Received: 29 March 2022 | Revised: 11 July 2022 | Accepted: 7 November 2022

DOI: 10.4218/etrij.2022-0115

ORIGINAL ARTICLE

ETRI Journal WILEY

Task offloading under deterministic demand for vehicular edge computing

Haotian Li[1] | **Xujie Li**[1,2,3] | **Fei Shen**[1]

[1]College of Computer and Information, Hohai University, Nanjing, China

[2]Key Laboratory of Wireless Sensor Network Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Science, Shanghai, China

[3]Nantong Ocean and Coastal Engineering Research Institute, Hohai University, Nantong, China

Correspondence
Xujie Li, College of Computer and Information, Hohai University, Nanjing, China.
Email: lixujie@hhu.edu.cn

Funding information
This research was supported by the Fundamental Research Funds for the Central Universities (B220203034 and B200203131) & the Postgraduate Research and Practice Innovation Program of Jiangsu Province (KYCX20-0537), in part by Future Network Scientific Research Fund Project (FNSRFP-2021-YB-7), in part by Open Research Fund Key Laboratory of Wireless Sensor Network and Communication of Chinese Academy of Sciences (20190914), in part by Water Science and Technology Program of Jiangsu (2020028), and in part by Social and People's Livelihood Technology in Nantong City (MS22021042).

Abstract
In vehicular edge computing (VEC) networks, the rapid expansion of intelligent transportation and the corresponding enormous numbers of tasks bring stringent requirements on timely task offloading. However, many tasks typically appear within a short period rather than arriving simultaneously, which makes it difficult to realize effective and efficient resource scheduling. In addition, some key information about tasks could be learned due to the regular data collection and uploading processes of sensors, which may contribute to developing effective offloading strategies. Thus, in this paper, we propose a model that considers the deterministic demand of multiple tasks. It is possible to generate effective resource reservations or early preparation decisions in offloading strategies if some feature information of the deterministic demand can be obtained in advance. We formulate our scenario as a 0-1 programming problem to minimize the average delay of tasks and transform it into a convex form. Finally, we proposed an efficient optimal offloading algorithm that uses the interior point method. Simulation results demonstrate that the proposed algorithm has great advantages in optimizing offloading utility.

KEYWORDS
delay, task offloading, vehicle edge computing, wireless network

1 | INTRODUCTION

Vehicular edge computing (VEC), which can realize resource integration and data sharing at the edge of vehicular networks [1–3], faces many new challenges, for example, complicated computing resource management processes for task offloading and the unprecedented time-varying topology of vehicular networks [4]. Furthermore, tasks usually densely appear at different times instead of simultaneously arriving, which adds difficulty to scheduling and resource waste or task latency due to continual decision making once a task arrives.

VEC has realized powerful applications in vehicles with limited computation resources. For example, Wang and others [5] proposed a permissioned vehicular blockchain in VEC called Parking-chain, where parked vehicles can share idle computational resources with service requesters (SR). In addition, Lu and others [6] proposed a wireless digital dual-edge network model that integrates a digital dual-edge network with an edge network to realize new functionalities, for example, hyper-connected experiences and low-latency edge computing.

However, these VEC schemes, in which traditional edge or cloud servers typically provide computational resources to requesting vehicles, do not consider another feasible case that some vehicles can also share their idle computational resources with edge SR [3, 7–9]. In fact, many sensors, for example, traffic cameras, regularly upload the data they acquire to the transportation system in a practical traffic environment. As a result, the data characteristics (i.e., data size and computational load) collected by such a sensor in a given period may not have huge differences compared with those with other periods. Thus, deterministic demand could be obtained in advance, which can be used to improve resource management. Additionally, these tasks may appear at different times with tiny time intervals in a subsequent short period (e.g., a few minutes or several seconds) with their determined characteristic information (e.g., generated time, data size, and computational load) could be obtained in advance. For example, some sensors (e.g., cameras) continuously collect data over a given period (e.g., 30 s or 60 s) and then periodically pack their data into a task package every 30 s or 60 s. In addition, such sensor nodes inform a control center about the deterministic demand before packages are completely generated, which means that the deterministic demand information can be utilized early. Thus, the predetermined information must be considered in resource management processes, which is expected to promote the design of effective offloading strategies.

In other works, multiple tasks typically appear at the same time and are allocated simultaneously to edge severs; however, this ignores the asynchronism of the task appearance time. In practical applications, various sensors or equipment work independently with different assignments and potentially collect or process different types of data with different data sizes. Under such conditions, the period of data collection would differ from each other for different sensors. As a result, sensors tend to complete their data collection processes at different times, which introduces complications in terms of realizing effective task offloading. Thus, in this paper, we exploit deterministic demand information because such information can be obtained in advanced due to the periodical work characteristics of sensors. In other words, the demand for computation resources for tasks can be predetermined before these data are finally packaged by sensors.

In this case, deterministic demand information can be utilized to solve the asynchronism problem in task appearance time. Thus, it would make appropriate offloading plans quickly according to the deterministic demand, which contributes to a reasonable and efficient allocation of computational resources.

To address these issues, we first propose a practical task offloading model that utilizes the deterministic demand of multiple tasks. We then introduce a discrete-time system to forecast the mobility of vehicles and formulate a typical 0-1 programming problem, which is NP hard, based on the average task delay. Finally, an algorithm is designed to address this optimization problem and simulation results are presented to validate the convergence and effectiveness of the proposed algorithm. Our primary contributions are summarized as follows.

1. We propose a practical task offloading model that considers the deterministic demand of tasks generated by sensors that regularly upload data.
2. We introduce a discrete-time system to forecast vehicle mobility, and we formulate this model as a typical 0-1 programming problem, which is NP hard, according to the average task delay.
3. An algorithm is designed to address the 0-1 optimization problem, and then simulation results are presented and discussed to validate the convergence and effectiveness of the proposed method.

The remainder of this paper is organized as follows. Section 2 introduces the system model for the target scenario. Section 3 describes the problem formulation and presents an effective algorithm for the formulated problem. Section 4 presents simulation results and present an accompanying analysis. Finally, the paper is concluded in Section 5.

2 | SYSTEM MODEL

Figure 1 shows the framework for upcoming queued task offloading in a VEC network. Here, some roadside units (RSUs) distributed uniformly along the road, which are denoted $k \in \mathcal{K} = \{1, \ldots, K\}$, and there are also some smart vehicles, which are denoted $i \in \mathcal{I} = \{1, \ldots, N\}$, driving on the road at different speeds. These RSUs are connected via fibers, and their data are shared. Here, for a given period, if the servers of the RSUs are busy and overloaded or the CPUs are overwhelmed, the servers of the RSUs

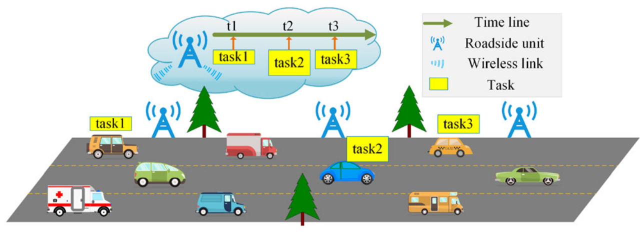


FIGURE 1 System model

can request edge services from vehicles on the road with idle computational resources.

Assume that in an upcoming period of finite time (likely several seconds), the infrastructures (RSUs) are expected to generate a series of subsequent tasks along the timeline, denoted $j \in \mathcal{J} = \{1, \ldots, M\}$. These tasks will appear at different times, denoted $t_j \in \mathcal{T} = \{t_1, \ldots, t_M\}$, which satisfies $t_1 \le t_2 \ldots \le t_j \ldots \le t_M$, and the tasks can be characterized by (t_j, C_j, D_j), where C_j and D_j are the computational density and data size of a task, respectively.

2.1 | Mobility model

Because the highly time-varying vehicular state information, we can use a discrete-time system to formulate the vehicular state. Here $s \in \mathcal{S}$ denotes the index of time slots, and each time slot duration $\Delta\tau$ is sufficiently small. Thus, we assume that vehicular state information loc_i remains unchanged, which can be expressed as follows:

$$loc_i^{\text{xaxis}}(s+1) = loc_i^{\text{xaxis}}(s) + \Delta\tau v_i(s), \tag{1}$$

where loc_i^{xaxis} is the coordinate location of vehicle i at its running direction. The lateral location changes finitely compared with the running direction; thus, we approximately regard the lateral location as a constant. Here, $k \in \mathcal{K} = \{1, \ldots, K\}$ represents the index of RSUs, and the distance between vehicle i and RSU k is expressed as follows:

$$L_{i,k}[s] = \sqrt{\left(loc_i^{\text{xaxis}}[s] - loc_k^{\text{RSU}}\right)^2 + \left(loc_i^{\text{yaxis}}[s]\right)^2}, \tag{2}$$

During the offloading process, vehicles may experience several channel switches (from the last RSU's coverage area to the next) due to their mobility. In a certain cell, the ideal access point for vehicles is the RSU that covers this cell. Thus, the ideal distance for communication between vehicles and RSUs is given as follows:

$$L_i^{\text{com}}[s] = \min_{k \in \mathcal{K}} \{L_{i,k}[s]\}. \tag{3}$$

Using an orthogonal communicate mode, user vehicles would be interfered by the signals from adjacent vehicles, and hence, it is essential to consider such interference. Thus, the communication capacity between RSU i and vehicle j is expressed as follows:

$$R_{i,j}[s] = B \log\left(1 + \frac{P^{\text{RSU}}\left(L_i^{\text{com}}[s]\right)^{-\alpha}}{N_0 + \sum\limits_{m \in \mathcal{J}, m \ne i} P^{\text{RSU}}\left(L_m^{\text{com}}[s]\right)^{-\alpha}}\right). \tag{4}$$

$$Sw_{i,j}^{up} = \begin{cases} 0, \text{ when } \underset{k \in \mathcal{K}}{arg\ \min}\left\{L_{i,k}\left[t_j + T_j^{\max}\right]\right\} = 1, i \in \mathcal{I} \\ \dfrac{1}{2L^{\text{cell}}}\left\{ loc^{\text{RSU}}_{\underset{k \in \mathcal{K}}{arg\min}\left\{L_{i,k}\left[t_j + T_j^{\max}\right]\right\}} + loc^{\text{RSU}}_{\underset{k \in \mathcal{K}}{arg\min}\left\{L_{i,k}\left[t_j + T_j^{\max}\right]\right\}-1} \right. \\ \left. -\left(loc_{arg\min_{k \in \mathcal{K}}\left\{L_{i,k}\left[t_j\right]\right\}}{}^{\text{RSU}} + loc^{\text{RSU}}_{\underset{k \in \mathcal{K}}{arg\min}\left\{L_{i,k}\left[t_j\right]\right\}+1} \right) \right\} + 1 \\ , \text{ when } \underset{k \in \mathcal{K}}{arg\ \min}\left\{L_{i,k}\left[t_j + T_j^{\max}\right]\right\} \ge 2, i \in \mathcal{I}. \end{cases} \tag{5}$$

Here, $L^{\max,\text{com}} = \frac{\sqrt{2}}{2} L^{\text{cell}}$ is the maximum communication distance, where L^{cell} is the length of a cell in *xaxis*. Thus, the minimum communication rate is $R_{\min} = B \log\left(1 + P^{\text{RSU}} N_0^{-1} (L^{\max})^{-\alpha}\right)$, which is a lower bound. Therefore, the maximum transmission delay for task j in the given time slots is $T_j^{\max} = D_j (R_{\min} \Delta\tau)^{-1}$.

The mobility of vehicles has a considerable influence on task offloading problems due to the time-varying positions of the vehicles. During this process, the distances between vehicles and RSUs may change dramatically, which can affect the communication between vehicles and RSUs. In addition, channel switches may occur several times among different communication cells, which can result in additional delays. Thus, the transmission delay of tasks is affected by the mobility of vehicles, and the task offloading problem primarily depends on transmission and computing delays. In this case, the vehicle mobility model must be considered when generating allocation strategies for task offloading.

In addition, during this process, vehicles may periodically pass through several cells and experience channel switches, which may result in link instability and the additional overhead of delay (Δcs represents the delay for a single switch). During the process for vehicle i receiving task j, the upper bound of the switch times is given in (5). We then obtain the following:

$$k_i[s]=\begin{cases}1,\text{if}\\ \left(loc_i^{\text{xaxis}}(s)-\dfrac{\left(loc_k^{\text{RSU}}+loc_{k+1}^{\text{RSU}}\right)}{2}\right)\times\\ \left(loc_i^{\text{xaxis}}(s+1)-\dfrac{\left(loc_k^{\text{RSU}}+loc_{k+1}^{\text{RSU}}\right)}{2}\right)\leq 0\\ 0,\text{else.}\end{cases} \tag{6}$$

Here, $k_i'[s]$ is used to mark the time slot when vehicle i passes through the boundary between two cells. If $k_i'[s]=1$, vehicle i reaches the boundary of the two cells (and vice versa). Then, let $T_{i,j}^{Sw,k}=s, if\, k_i'[s]=1$ for $k\in\mathcal{K}$ and assume $T_{i,j}^{Sw,k''-1}<t_j<T_{i,j}^{Sw,k''}$. Thus, the assisted parameter ϖ_j is followed by

$$\begin{aligned}\varpi_{i,j}=\arg\min_s\Bigg|&\sum_{s=t_j}^{T_{i,j}^{Sw,k''}}R_{i,j}[s]\Delta\tau\\ &+\sum_{s=T_{i,j}^{Sw,k''}+\Delta cs}R_{i,j}[s]\Delta\tau+\cdots\\ &+\sum_{s=T_{i,j}^{Sw,k''-1+Sw_{i,j}^{up}}+Sw_{i,j}\Delta cs}^{T_{i,j}^{Sw,k''++1}}R_{i,j}[s]\Delta\tau-D_j\Bigg|.\end{aligned} \tag{7}$$

According to the ϖ_j obtained in (7), we can obtain the actual transmission delay using (8).

2.2 | Offloading and computing delay

Here, there is only a communication link for task offloading between each vehicle and the RSUs, and once a task appears, it is transmitted to a selected vehicle offering service. The transmission delay and computing delay for vehicle i for task j are denoted $T_{i,j}^t$ and $T_{i,j}^c=C_jD_j/f_{i,j}$, respectively. The computing frequency $f_{i,j}$ is the computational resources (i.e., CPU cycles) offered by vehicle j to serve task i. We consider that $f_{i,j}$ follows a uniform distribution from 0–100 (in CPU cycles), as shown in Table 1. Introducing decision variables $x_{i,j}=\{0,1\}\in\mathbf{X}$ (if $x_{i,j}=1$, vehicle i serves task j, and vice versa), and the task delay (only for $j=1$) is expressed as follows:

$$T_j=\sum_{i\in\mathcal{I}}x_{i,j}\left(T_{i,j}^t+T_{i,j}^c\right). \tag{9}$$

At this point, the transmission delay of task j is equal to $T_{i,j}^r$, which is the time consumption from t_j to vehicle i successfully receiving the package, that is, $T_{i,j}^r=T_{i,j}^t$. Thus, if $j\geq 2$, it is possible for task u ($u<j, u\in\mathcal{J}$) to wait for transmission because task $j-1$ still occupies the channel of vehicle i, which can be given as $t_u+T_{i,u}^r-t_j>0$. Then, the initial time t_j in (5)–(8) must be replaced by $t_j+\max_{g\in\mathcal{G}}\left\{x_{i,g-1}\left(t_{g-1}+T_{i,g-1}^r-t_j\right),0\right\}$, and $T_{i,j}^r$ is obtained as follows:

$$T_{i,j}^r=\max_{g\in\mathcal{G}}\left\{x_{i,g-1}\left(t_{g-1}+T_{i,g-1}^r-t_j\right),0\right\}+T_{i,j}^t, \tag{10}$$

where $\mathcal{G}=\{2,\ldots,j\}$. If $j\geq 2$, it is possible for vehicle i to serve more than one task. Here, we assume a first come, first served prioritization of tasks. Thus, there are two cases for T_j. In the first case, that is, $(t_u+T_u\leq t_j, u<j)$,

$$T_{i,j}^t=\begin{cases}\varpi_{i,j}-t_j,\text{ when}\left(\sum_{s=t_j}^{T_{i,j}^{Sw,k''}}R_{i,j}[s]\Delta\tau+\sum_{s=T_{i,j}^{Sw,k''}+\Delta cs}^{T_{i,j}^{Sw,k''+1}}R_{i,j}[s]\Delta\tau+\cdots+\sum_{s=T_{i,j}^{Sw,k''-1+Sw_{i,j}^{up}}+Sw_{i,j}\Delta cs}^{T_{i,j}^{Sw,k''+Sw_{i,j}^{up}}}R_{i,j}[s]\Delta\tau-D_j\right)\geq 0\\ \varpi_{i,j}+1-t_j,\text{ when}\left(\sum_{s=t_j}^{T_{i,j}^{Sw,k''}}R_{i,j}[s]\Delta\tau+\sum_{s=T_{i,j}^{Sw,k''}+\Delta cs}^{T_{i,j}^{Sw,k''+1}}R_{i,j}[s]\Delta\tau+\cdots+\sum_{s=T_{i,j}^{Sw,k''-1+Sw_{i,j}^{up}}+Sw_{i,j}\Delta cs}^{T_{i,j}^{Sw,k''+Sw_{i,j}^{up}}}R_{i,j}[s]\Delta\tau-D_j\right)<0.\end{cases} \tag{8}$$

TABLE 1 Simulation parameters

Parameters	Description	Value
W	Bandwidth of RSUs for each vehicle	1 MHz
α	Pass-loss exponent of communication	4
N_0	Noise power	10^{-15} W
P^{RSU}	Transmitted power of RSUs	2 W
w_{lane}	Width of each lane	3.6 m
L_{cell}	Length of a cellular covered by RSUs	100 m
M	Number of tasks	2–8
D_j	Data size of task j	50 MB–400 MB
C_j	Computing density of tasks	50–400
$f_{i,j}$	Computation resources of vehicles	U(0, 100)
N	Number of vehicles	5–40
K	Number of RSUs	20
v	Speed of vehicles	10 m/s–20 m/s
s	Length of a time slot	0.01 s

task j has no additional time consumption except waiting for transmission; therefore, T_j is similar to (9), which is $T_j = \sum_{i \in \mathcal{N}} x_{i,j}\left(T^r_{i,j} + T^c_{i,j}\right)$. In the second case, that is, $(t_u + T_u > t_j, u < j)$, it is possible for vehicle i to wait for computing because the vehicular CPU is occupied by previous tasks, and the time consumption can be expressed as follows:

$$T^{\mathrm{wait}}_{i,j} = \max\left\{\max_{g \in \mathcal{G}}\left\{x_{i,g-1}\left(t_{g-1} + T_{g-1} - t_j\right)\right\}, T^r_{i,j}\right\}. \quad (11)$$

In this case, T_j is followed by

$$T_j = \sum_{i \in \mathcal{I}} x_{i,j}\left(T^{\mathrm{wait}}_{i,j} + T^c_{i,j}\right). \quad (12)$$

Then, the delay of task j can be obtained as follows.

$$T_j = \begin{cases} \sum_{i \in \mathcal{I}} x_{i,j}\left(T^t_{i,j} + T^c_{i,j}\right), \text{when} j = 1 \\ \sum_{i \in \mathcal{I}} x_{i,j}\left(T^r_{i,j} + T^c_{i,j}\right) \\ \quad , \text{when} j \geq 2 \text{ and } t_{j-1} + T_{j-1} \leq t_j \\ \sum_{i \in \mathcal{I}} x_{i,j}\left(T^{\mathrm{wait}}_{i,j} + T^c_{i,j}\right) \\ \quad , \text{when } j \geq 2 \text{ and } t_{j-1} + T_{j-1} > t_j \end{cases} \quad (13)$$

$$\begin{aligned} \mathbf{P1} \quad \min_{x_{i,j} \in \mathbf{X}} \frac{1}{M}\Bigg\{ & \sum_{j=1}\sum_{i \in \mathcal{I}} x_{i,j}\left(T^t_{i,j} + T^c_{i,j}\right), \\ & + \sum_{t_{j-1}+T_{j-1} \leq t_j}^{j \geq 2} \sum_{i \in \mathcal{I}} x_{i,j}\left(\max_{g \in \mathcal{G}}\left\{x_{i,g-1}\left(t_{g-1} + T^r_{i,g-1} - t_j\right), 0\right\}\right. \\ & \left. + T^t_{i,j} + T^c_{i,j}\right) + \sum_{t_{j-1}+T_{j-1} > t_j}^{j \geq 2} \sum_{i \in \mathcal{I}} x_{i,j} \\ & \left(\max\left\{\max_{g \in \mathcal{G}}\left\{x_{i,g-1}\left(t_{g-1} + T_{g-1} - t_j\right)\right\}, T^r_{i,j}\right\} + T^c_{i,j}\right)\Bigg\} \end{aligned} \quad (14)$$

$$s.t. x_{i,j} \in \{0,1\}, i \in \mathcal{I}, j \in \mathcal{J}, \quad (14a)$$

$$\sum_{i \in \mathcal{J}} x_{i,j} = 1, j \in \mathcal{J}. \quad (14b)$$

3 | PROBLEM FORMULATION AND PROPOSED ALGORITHM

The proposed optimal edge computing problem, which minimizes the average delay of tasks in queue, can be formulated as **P1**, as shown in (14).

Note that problem **P1** is NP hard under constraints (14a) and (14b); thus, binary variable $x_{i,j}$ can be slacked into $y_{i,j} \in [0,1]$. As a result, this problem can be transformed as follows.

$$\mathbf{P2} \qquad \min_{y_{i,j} \in \mathbf{Y}} f(\mathbf{Y}), \quad (15)$$

$$s.t. y_{i,j} - 1 \leq 0, i \in \mathcal{I}, j \in \mathcal{J}, \quad (15a)$$

$$-y_{i,j} \leq 0, i \in \mathcal{I}, j \in \mathcal{J}, \quad (15b)$$

$$\sum_{i \in \mathcal{I}} y_{i,j} - 1 = 0, j \in \mathcal{J}. \quad (15c)$$

This problem can be approached as a linear programming problem, and we consider solving it using the interior point method [10]. First, constraint (15c) can be further slacked into $\sum_{i \in \mathcal{I}} y_{i,j} - 1 \leq 0$; therefore, the penalty function can be formulated as follows:

$$\begin{aligned} F(\mathbf{Y}) = f(\mathbf{Y}) - \frac{1}{\alpha}\sum_{j \in \mathcal{J}} \log\left(1 - \sum_{i \in \mathcal{I}} y_{i,j}\right) \\ -\frac{1}{\alpha}\sum_{i \in \mathcal{I}}\sum_{j \in \mathcal{J}}\left(\log\left(1 - y_{i,j}\right) + \log y_{i,j}\right), \end{aligned} \quad (16)$$

where

$$\nabla_{y_{i,j}}F(\mathbf{Y})=\begin{cases}\frac{1}{M}\left(T^t_{i,j}+T^c_{i,j}\right)+\frac{1}{\alpha}\frac{1}{\left(1-\sum_{i\in\mathcal{I}}y_{i,j}\right)}-\frac{1}{\alpha}\left(\frac{1}{y_{i,j}}-\frac{1}{1-y_{i,j}}\right), & \text{if } j=1\\ \frac{1}{M}(\max_{g\in\mathcal{G}}\left\{y_{i,g-1}\left(t_{g-1}+T^r_{i,g-1}-t_j\right)0\right\}+T^t_{i,j}+T^c_{i,j})+\frac{1}{\alpha}\frac{1}{\left(1-\sum_{i\in\mathcal{I}}y_{i,j}\right)}-\frac{1}{\alpha}\left(\frac{1}{y_{i,j}}-\frac{1}{1-y_{i,j}}\right), & \\ \qquad \text{if } j\geq 2 \text{ and } t_{j-1}+T_{j-1}\leq t_j & \\ \frac{1}{M}(\max\{\max_{g\in\mathcal{G}}\left\{y_{i,g-1}(t_{g-1}+T_{g-1}-t_j)\right\},T^r_{i,j}\}+T^c_{i,j})+\frac{1}{\alpha}\frac{1}{\left(1-\sum_{i\in\mathcal{I}}y_{i,j}\right)}-\frac{1}{\alpha}\left(\frac{1}{y_{i,j}}-\frac{1}{1-y_{i,j}}\right), & \\ \qquad \text{if } j\geq 2 \text{ and } t_{j-1}+T_{j-1}> t_j. & \end{cases} \tag{17}$$

$\alpha>0$ is the penalty factor, and $F(\mathbf{Y})$ will be more approximate to the original function as α increases. We can then obtain the gradient of $F(\mathbf{Y})$, which is shown in (17).

Based on $\nabla_{y_{i,j}}F(\mathbf{Y})$, we obtain a set of solutions to $\min F(\mathbf{Y})$ with the gradient method and then transform it to satisfy constraint (15c). However, the original problem is a 0-1 programming problem; thus, we must transform the continuous solution $\mathbf{Y}$ into an integer solution $\mathbf{X}$. Here, the main steps of the proposed edge computing scheme are summarized in Algorithm 1, which is referred to as the TODD algorithm.

Algorithm 1 Algorithm for problem (14) (TODD)

```
Require: Penalty factor $\alpha$, vehicular status information:
    original position, speed, computation resource, etc.
Ensure: Optimal solution $\mathbf{X}$.
1:  Initialization: Set an initial matrix $\mathbf{Y}$ for problem 15.
2:  Obtain the slack form of primary problem using $\mathbf{Y}$ according to 15;
3:  Construct penalty function $F(\mathbf{Y})$, and $\sum_{i\in\mathcal{I}} y_{i,j}-1=0, j\in\mathcal{J}$ should be further relaxed to $\sum_{i\in\mathcal{I}} y_{i,j}-1\leq 0, j\in\mathcal{J}$, according to 16;
4:  repeat
5:      Update $\mathbf{Y}$ with $y_{i,j}=\frac{y_{i,j}}{\sum_{i\in\mathcal{I}} y_{i,j}}$, to meet the constrain $\sum_{i\in\mathcal{I}} y_{i,j}-1=0, j\in\mathcal{J}$;
6:      Update $T^t_{i,j}$ and $T^r_{i,j}$ using $\mathbf{Y}$, according to (5)–(8), and with $t_j$ replaced by $t_j+\max_{g\in\mathcal{G}}\left\{y_{i,g-1}\left(t_{g-1}+T^r_{i,g-1}-t_j\right),0\right\}$;
7:      Update $\mathbf{Y}$ based on gradient method according to $y_{i,j}=y_{i,j}-b\nabla_{y_{i,j}}F(\mathbf{Y})$;
8:      if $y_{i,j}=\max_{i\in\mathcal{I}}\left\{y_{i,j}\right\}$ then
9:          $x_{i,j}=1$;
10:     else
11:         $x_{i,j}=0$;
12:     end if
13:     Obtain the objective function values according to formula (14), and choose the better one compared with the previous iteration;
14: until $f^{\text{iteration}}(\mathbf{X})-f^{\text{iteration}+20}(\mathbf{X})\leq 10^{-3}$.
```

4 | SIMULATION AND DISCUSSION

In this section, we present simulation results to evaluate the effectiveness of the proposed algorithm. Here, we consider a straight three-lane road with RSUs evenly located in each area and 10 vehicles driving at different speeds, that is, 20 m/s, 15 m/s, and 10 m/s. In this simulation, we set three tasks with different data sizes (100 MB, 200 MB, and 300 MB) and computational densities (100, 200, and 300), and these tasks appear at different times under a Poisson distribution.

The unit of computational density C_j is CPU cycles per bit, and the data size unit D_j is bits. Thus, the total required CPU cycles to execute a task is given by C_jD_j. In addition, the unit of computational resources is the CPU cycles offered by the edge servers in 1 s [8].

In this simulation, we evaluated four different algorithms or schemes as benchmarks to verify the effectiveness of the proposed algorithm. These benchmark algorithms/schemes are summarized as follows.

1. TODD, the proposed algorithm "**T**ask **O**ffloading under **D**eterministic **D**emand."
2. Genetic algorithm (GA). The GA is a classic optimization algorithm that is widely used in various fields. Here, we have 10 groups, and we used the mutation, crossover, and selection operations to identify good solutions during the process of the GA algorithm.
3. Folo is an algorithm proposed in Zhu and others [9].
4. The exhaustive method lists all feasible solutions and selects the best one.
5. The resource-greedy RG) method always selects the vehicle with the most computational resources.

Figure 2 shows the convergence of the proposed TODD algorithm implemented in a simulation environment compared with the benchmark schemes, that is, the GA algorithm, Folo [9], and the exhaustive method. As can be seen, satisfactory performance was obtained by the proposed algorithm in four iterations, and then it converged at approximately 34 iterations, which is very close

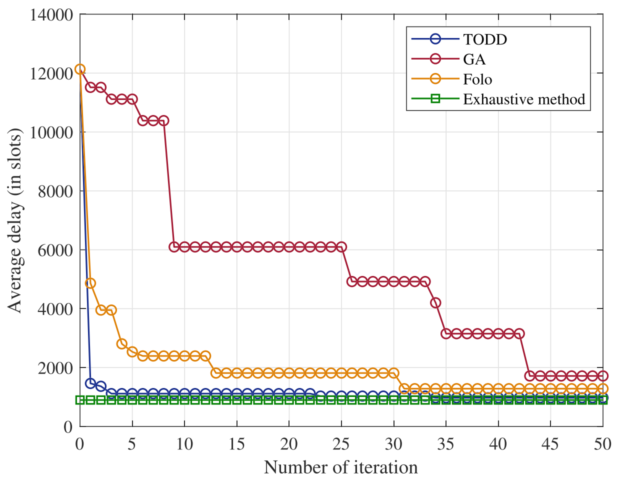


FIGURE 2 Process of iteration

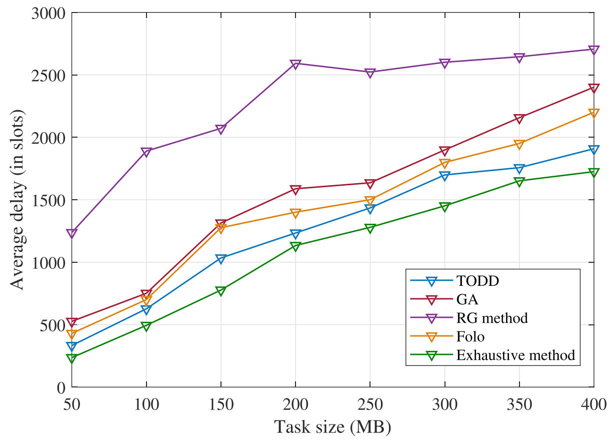


FIGURE 4 Performance comparison with different task sizes

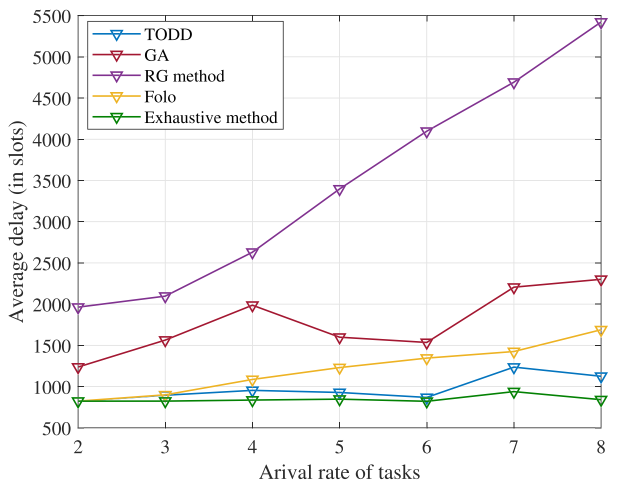


FIGURE 3 Performance comparison with different task arrival rates

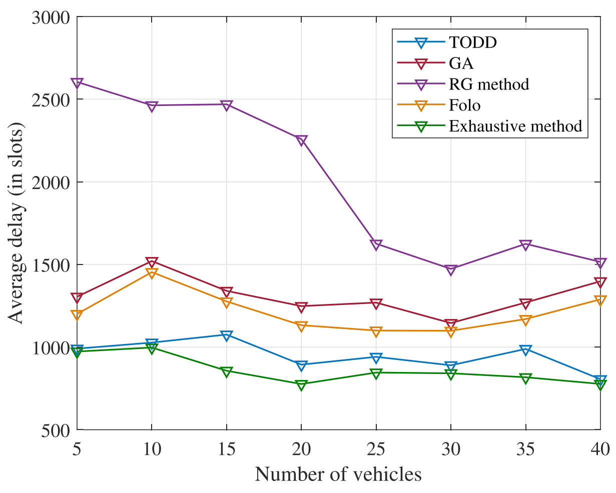


FIGURE 5 Performance comparison with different numbers of vehicles

to the results of the exhaustive method and remarkably better than the GA and Folo methods.

Figure 3 compares the utility relative to the different arrival rates of tasks with other edge caching scheduling methods. Here, the data size of the task flow was set to (200, 200) MB, (200, 200, 200) MB, and so on, and the task arrival/appearance time was determined according to the Poisson distribution. As can be seen, all schemes exhibited various trends as the arrival rates of tasks increased, where the proposed TODD algorithm is better than the GA, Folo, and RG method, which always selects the vehicle with the most available computational resource.

Figure 4 shows the effectiveness of the proposed method under different task sizes compared with other edge caching scheduling methods. Here, we have four tasks with average data sizes of 50–400 MB. As can be seen, all compared schemes exhibited various degrees of increasing rising trend as the task arrival rate increased. We found that the proposed TODD algorithm outperformed the GA, Folo, and RG methods.

Figure 5 verifies the effectiveness of the proposed TODD algorithm in terms of different numbers of vehicles compared with other edge caching scheduling schemes. Here, we set three tasks with an average size of 200 MB and various vehicle numbers ranging from 5 to 40. As can be seen, the proposed TODD algorithm outperformed the GA, Folo, and RG methods, and all methods exhibited various degrees of declining trend as the number of vehicles increased because a large number of users may bring large computation resources and choices.

5 | CONCLUSION

In this paper, we investigated a vehicle edge computing network with tasks that arrive at different times, which utilizes the deterministic demand of tasks. We constructed an optimization problem based on the system model and designed an efficient algorithm to solve the target problem. The performance of the proposed TODD algorithm was evaluated and compared with existing methods. The simulation results demonstrate that the edge system utility gained by the proposed algorithm was greater than that of the other two benchmark approaches. However, in complex traffic environments with highly dynamic changes in vehicle topology, the VEC network and task offloading strategies require further investigation. In addition, the age of information is also a potential direction for future work.

ACKNOWLEDGMENTS

This work was supported by Fundamental Research Funds for the Central Universities (B220203034 and B200203131) & the Postgraduate Research and Practice Innovation Program of Jiangsu Province (KYCX20-0537), in part by Future Network Scientific Research Fund Project (FNSRFP-2021-YB-7), in part by Open Research Fund Key Laboratory of Wireless Sensor Network and Communication of Chinese Academy of Sciences (20190914), in part by Water Science and Technology Program of Jiangsu (2020028), and in part by Social and People's Livelihood Technology in Nantong City (MS22021042).

CONFLICT OF INTEREST STATEMENT

The authors declare no potential conflict of interests.

ORCID

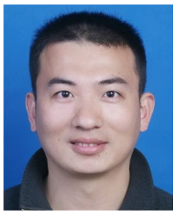
Haotian Li https://orcid.org/0000-0002-8939-7379
Xujie Li https://orcid.org/0000-0001-5486-5702
Fei Shen https://orcid.org/0000-0003-3797-2412

REFERENCES

1. K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, *Digital twin empowered content caching in social-aware vehicular edge networks*, IEEE Trans. Comput. Soc. Syst. **9** (2022), no. 1, 239–251.
2. W. Sun, P. Wang, N. Xu, G. Wang, and Y. Zhang, *Dynamic digital twin and distributed incentives for resource allocation in aerial-assisted internet of vehicles*, IEEE Internet Things J. **9** (2021), no. 8, 5839–5852. https://doi.org/10.1109/JIOT.2021.3058213
3. S. Zhou, Y. Sun, Z. Jiang, and Z. Niu, *Exploiting moving intelligence: Delay-optimized computation offloading in vehicular fog networks*, IEEE Commun. Mag. **57** (2019), no. 5, 49–55.
4. K. Zhang, J. Cao, and Y. Zhang, *Adaptive digital twin and multiagent deep reinforcement learning for vehicular edge computing and networks*, IEEE Trans. Industr. Inform. **18** (2022), no. 2, 1405–1413.
5. S. Wang, D. Ye, X. Huang, R. Yu, Y. Wang, and Y. Zhang, *Consortium blockchain for secure resource sharing in vehicular edge computing: a contract-based approach*, IEEE Trans. Netw. Sci. Eng. **8** (2021), no. 2, 1189–1201.
6. Y. Lu, S. Maharjan, and Y. Zhang, *Adaptive edge association for wireless digital twin networks in 6G*, IEEE Internet Things J. **8** (2021), no. 22, 16219–16230.
7. H. Li, X. Li, and W. Wang, *Joint optimization of computation cost and delay for task offloading in vehicular fog networks*, Trans. Emerg. Telecommun. Technol. **31** (2020), no. 2, 35–54.
8. Y. Sun, X. Guo, J. Song, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, *Adaptive Learning-Based Task Offloading for Vehicular Edge Computing Systems*, IEEE Trans. Veh. Technol. **68** (2019), no. 4, 3061–3074.
9. C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, *Folo Latency and Quality Optimized Task Allocation in Vehicular Fog Computing*, IEEE Internet Things J. **6** (2019), no. 3, 4150–4161.
10. S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, 2004.

AUTHOR BIOGRAPHIES

Haotian Li received the BSc degree in communication and information engineering from Hohai University, China, in 2018, where he is currently pursuing the PhD degree. His current research interests include mobile edge computing, internet of vehicles, reconfigurable intelligent surfaces and deep reinforcement learning.



Xujie Li received PhD degree in communication engineering from National Mobile Communications Research Lab., Southeast University in 2012. From 2016 to 2017, he was a visiting associate professor with the University of Warwick, UK. Currently, he is working as an associate professor in the College of Computer and Information Engineering in Hohai University. He is an associate editor of the IEEE Access. He serves as a reviewer for many journals and conferences, such as IEEE JSAC, TCOM, TWC, TSP, IET Com, Globecom, and so on. His research interests include fog computing network, D2D communications, energy efficient wireless sensor networks, and small cell technique.

Fei Shen was born in Anhui Province, China, on June 5, 1994. He received PhD degree in Communication Engineering from Hohai University, Nanjing, China, in 2022. His research interests include intelligent control systems, nonlinear system, signal processing, deep reinforcement learning, and mobile edge computing.

How to cite this article: H. Li, X. Li, and F. Shen, *Task offloading under deterministic demand for vehicular edge computing*, ETRI Journal **45** (2023), 627–635. https://doi.org/10.4218/etrij.2022-0115