IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 17, 2019, accepted January 13, 2020, date of publication January 21, 2020, date of current version January 30, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2968465

Delay-Optimized V2V-Based Computation Offloading in Urban Vehicular Edge Computing and Networks

CHEN CHEN[1], (Senior Member, IEEE), LANLAN CHEN[1], LEI LIU[1], (Student Member, IEEE), SHUNFAN HE[2], XIAOMING YUAN[3], DAPENG LAN[4], AND ZHUANG CHEN[5]

[1]State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China
[2]College of Computer Science, South-Central University for Nationalities, Wuhan 430073, China
[3]Qinhuangdao Branch Campus, Northeastern University, Qinhuangdao 066004, China
[4]Department of Informatics, University of Oslo, 0316 Oslo, Norway
[5]School of Computer and Information Security, Guilin University of Electronic Technology, Guilin 541004, China

Corresponding author: Lei Liu (tianjiaoliulei@163.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFE0126000, in part by the National Natural Science Foundation of China under Grant 61571338, Grant U1636209, Grant 61672131, and Grant 61901367, in part by the Key Research and Development Plan of Shaanxi Province under Grant 2017ZDCXL-GY-05-01, Grant 2019ZDLGY13-04, and Grant 2019ZDLGY13-07, in part by the Xi'an Key Laboratory of Mobile Edge Computing and Security under Grant 201805052-ZD3CG36, in part by the Innovation Fund of Xidian University under Grant 5001-20109195456, in part by the National Nature Science Foundation of China under Grant 6190021180, in part by the Natural Science Foundation of China under Grant 61901367, and in part by the Natural Science Foundation of China under Grant 61901099.

ABSTRACT The Internet of Vehicles (IoV) is an emerging paradigm, driven by recent advancements in vehicular communications and networking. Meanwhile, the capability and intelligence of vehicles are being rapidly enhanced, and this will have the potential of supporting a plethora of new exciting applications, which will integrate fully autonomous vehicles, the Internet of Things (IoT), and the environment. In view of the delay-sensitive property of these promising applications, as well as the high expense by using infrastructures and roadside units (RSU), the task offloading among vehicles has gained enormous popularity considering its free-of-charge and timely response. In this paper, by utilizing the gathering period of vehicles in urban environment due to stopped by traffic lights or Area of Interest (AOI), a task offloading scheme merely relying on vehicle-to-vehicle (V2V) communication is proposed by fully exploring the idle resources of gathered vehicles for task execution. Through formulating the task execution as a Min-Max problem among one task and several cooperative vehicles, the task executing time is optimized with the Max-Min Fairness scheme, which is further solved by the Particle Swarm Optimization (PSO) Algorithm. Extensive simulation demonstrate that our model could well meet the delay requirement of delay-sensitive application by cooperative computing among vehicles.

INDEX TERMS Internet of vehicle, computing offloading, low latency, task allocation.

I. INTRODUCTION

With the rapid development of social economy and the process of industrialization, the number of vehicles has been growing rapidly in cities [1]–[5]. However, owing to limited capacity, roads become crowded, which leads to a series of issues, e.g., traffic accident, environmental pollution. Face these relentless challenges, Internet of Vehicle (IoV) emerges to fully use the information and communication technologies for achieving the coordinated development of human, vehicle, and environment, which can alleviate traffic congestion, enhance transportation efficiency [6]–[8].

The associate editor coordinating the review of this manuscript and approving it for publication was Lu Liu.

However, there are still some technical problems to be solved in the popularization and development of IoV [9], [10], such as intelligent information processing [11], [12], that is, filtering and processing data in IoV to fetch out useful information [6], [13], [14]. For the challenge of intelligent information processing, it can be mainly summarized into three points. First, the IoV is engaged to provide timely services relevant to driving safety and traffic efficiency, which

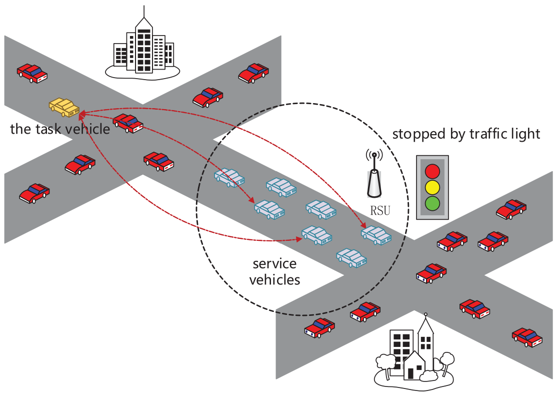


FIGURE 1. The scene of offloading computing.

requires strong capability of computing and intelligence on the edge [15]–[17]. Second, the strict latency requirement of applications often makes the V2V communication a pre-requisite, where a round-trip communication from infrastructure or RSU may fail the delay-sensitive services. Third, in view of the high expense by using infrastructure or RSU to process and store the collected information, the V2V communication is free-of-charge thus attracting more attentions. Considering above concerns, the information processing cooperated by surrounding vehicles in a V2V pattern, is very promising [2], [18], [19].

In addition, in urban environment, vehicles usually gather together for various reasons, such as waiting for traffic lights, passing through toll stations, or attracted by an AOI. For example, as shown in Fig.1, these service vehicles that are slowly moving together due to traffic lights will form a resource pool for a short period of time. These vehicles may have rich and idle resource for task execution, i.e., information processing in IoV [20]. Therefore, we can certainly offload the task to these service vehicles, and make the task vehicle and several surrounding vehicles cooperatively completing the vehicle-carried task to reduce the task execution time [21]–[23].

There has been considerable amount of work focusing on computation offloading [24]–[26]. The authors considered/analyzed the problem of binary offloading decision, where each user independently chooses whether to execute the task locally or to offload the whole task to the edge servers, to minimize the task execution time. Studies, such as Luo *et al.* [26] used dynamic programming and improved greedy algorithm to assign different tasks to multiple edge servers, to minimize the latency. Liu *et al.* [27] formulated the multiple vehicles computation offloading problem as a multi-user computation offloading game problem, prove the existence of Nash equilibrium (NE) of the game and propose a distributed computation offloading algorithm to compute the equilibrium. Compared to binary offloading, it is more reasonable to offload partially to service vehicles, since wireless channel resource is limited and local computing power should not be ignored [28]–[30]. Motivated by such considerations, in this paper, we dynamically offload parts of the computation task from the task vehicle to service vehicles [14]. What is more, in IoV, rapid vehicle movement results in frequent network topology changes and shorter communication link lifetime. So in the process of offloading, we need to consider the node movement problem to prevent communication link interruption [31].

Combine the above two considerations, in this paper, we propose a cooperative task scheduling scheme in which one task vehicle and multiple service vehicles jointly execute the vehicle-carried task [14], [32]. The task vehicle comprehensively considers the computing power and maximum service time of each service vehicle to decide what percentage of the task should be assigned to each service vehicle, to minimize the task execution time. Here, the maximum service time of each service vehicle depends not only on its residence time at the gathering point, but also on the relative movement of the task vehicle and the service vehicle. Because we must ensure that the task vehicle and each service vehicle are always within the communication range of each other. The key contribution of this paper include:

- We propose the concept of "resource pool" formed by service vehicles that are slowly moving together. Based on these idle resources that can be scheduled in resource pool, we propose a cooperative task scheduling scheme to minimize the task execution time.
- We formulate the task execution time optimization model as Min-Max problem under the permissible latency constraint. In addition, we also incorporate node mobility into problem formulation.
- For the Min-Max problem, we modify Max-Min Fairness Algorithm and Particle Swarm Optimization (PSO) Algorithm respectively according to whether to call all service vehicles, to find the best task allocation scheme to minimize the task execution time. Simulation results demonstrate the effectiveness of proposed schemes.

The rest of this paper is organized as follows. The system model which includes the mobility model, communication model, computation model and problem formulation is presented in Section II. The problem solution is given in Section III. We evaluate the performance of proposed schemes and provide illustrative results in Section IV. We conclude this paper in Section V. To make the readers easily follow, the notations used in our paper are listed in Table1.

II. SYSTEM MODEL

We mainly focus on V2V-based task offloading model in order to minimize the needed time for task processing by leveraging all available idle resources in vehicular networks as shown in Fig.2. The considered scenario includes one task vehicle which has one task to execute and N service vehicles which have idle computation resources, where N vehicles are within the communication range of task vehicle. Specially, let $N = \{Vidle_1, Vidle_2, Vidle_3 \cdots Vidle_N\}$ be the set of N service vehicles. We adopt three parameters $\{c, data, t^{\max}\}$ to indicate the generated task of task vehicle. Here, c (in CPU cycles per bit) is the computation resource

TABLE 1. List of important notations.

Parameter	Value
$N = \{Vidle_1, Vidle_2 \cdots Vidle_N\}$	The set of service vehicles
c	The amount of computing resource required for task processing
$data$	The data size of computation task
$t^{\max}$	The completion deadline for computation task
$b = \{b_1, b_2, b_3 \cdots b_N\}$	The task offloading strategy, b_i indicates the task segmentation ratio that service vehicle i undertake
$X_0(t)$	The position of the task vehicle at time t
$X_i(t)$	The position of the service vehicle i at time t
ΔX_i	The relative distance between the task vehicle and the service vehicle i
r_i	The uplink data rate between the task vehicle and the service vehicle i
f_0	The CPU computing capability of the task vehicle
f_i	The CPU computing capability of the service vehicle i
t_{local}	The task execution time locally
t_{edge}	The task execution time of computation offloading
t_i^c	The computation time of service vehicle i
t_i^{tr}	The communication time of service vehicle i
t_0	The computing time of task vehicle
t_i	The total time of service vehicle i
t_{res_i}	The residence time of service vehicle i
$t_{\max\_i}$	The maximum service time of service vehicle i

required for processing the task, *data* (in bits) specifies the task size, and $t^{\max}$ represents the completion deadline for task execution. Each service vehicle uploads its own information to its nearby RSU, including vehicle ID, position, velocity and available computation resource. The task vehicle can get the information about these service vehicles from this RSU. Generally, the task vehicle can process its task locally or by computation offloading. With the aim to reduce the task execution time, the task vehicle can offload part of its task to surrounding vehicles which possess idle resources. Then, two problems arise for the task vehicle: (1) how to select service vehicles for computation offloading, and (2) how to offload the computation task to these service vehicles.

A. MOBILITY MODEL

In our model, we assume that identical vehicles are driving on a two-way straight road, regardless of vehicle turning. Vehicles can move toward the oppositive directions. We adapt Intelligent Driver Model with Intersection Management (IDM_IM) mobility model to accurately capture the realistic mobility characteristics of moving vehicles [33], [34].

As for the behavior of a single car in straight road, it can be described as follows. Assume the length of vehicle α is l_α, the position of vehicle α at time t is $x_\alpha(t)$, the velocity of vehicle α at time t is $v_\alpha(t)$, the position of the leading vehicle $\alpha-1$ at time t is $x_{\alpha-1}(t)$, the velocity of vehicle $\alpha-1$ at time t is $v_{\alpha-1}(t)$. Note that the velocity here represents a vector. If it is less than 0, the vehicle is moving toward the left; otherwise, the vehicle is driving toward the right. The acceleration in the IDM model can obtained as follows:

$$\frac{dv_\alpha(t)}{dt} = a_\alpha[1 - (\frac{v_\alpha(t)}{v_\alpha^*})^\delta - (\frac{s^*(v_\alpha(t), \Delta v_\alpha(t))}{s_\alpha(t)})^2] \quad (1)$$

where a_α is maximum acceleration, δ is acceleration exponent, the gap $s_\alpha(t) := x_\alpha(t) - x_{\alpha-1}(t) - l_\alpha$, the velocity difference $\Delta v_\alpha(t) := v_\alpha(t) - v_{\alpha-1}(t)$, v_α^* is desired velocity for vanishing interactions($s_\alpha(t) \rightarrow \infty$). The desired minimum gap $s^*(v_\alpha(t), \Delta v_\alpha(t))$ can be obtained as follows:

$$\begin{aligned} s^*(v_\alpha(t), \Delta v_\alpha(t)) &= s'_\alpha + s''_\alpha\sqrt{\frac{v_\alpha(t)}{v_\alpha^*}} + v_\alpha(t)T \\ &\quad + \frac{v_\alpha(t)\Delta v_\alpha(t)}{2\sqrt{a_\alpha b_\alpha}} \end{aligned} \quad (2)$$

where s'_α and s''_α are jam distances, T is safe time headway and b_α is desired deceleration [35].

According to [35], we usually set $s''_\alpha = 0$, so we can get

$$s^*(v_\alpha(t), \Delta v_\alpha(t)) \approx s'_\alpha + v_\alpha(t)T + \frac{v_\alpha(t)\Delta v_\alpha(t)}{2\sqrt{a_\alpha b_\alpha}} \quad (3)$$

According to [36], we can integrate Eq.(1) with a simple Euler scheme using a coarse time discretization of $\Delta t = 0.4s$, and get velocity and displacement calculation formula:

$$\begin{aligned} v_\alpha(t+\Delta t) &= v_\alpha(t) + \Delta t \cdot \frac{dv_\alpha(t)}{dt} \\ x_\alpha(t+\Delta t) &= x_\alpha(t) + \Delta t \cdot v_\alpha(t) + \frac{1}{2}(\frac{dv_\alpha(t)}{dt})(\Delta t)^2. \end{aligned} \quad (4)$$

Here, smaller value of Δt yields nearly indistinguishable results [36]. By partitioning the interval $[t, t+t_\alpha]$ based on step Δt, we can get the displacement $\Delta X_\alpha(t_\alpha)$ as follows:

$$\begin{aligned} &\Delta X_\alpha(t_\alpha) \\ &\quad = x_\alpha(t+t_\alpha) - x_\alpha(t) \\ &\quad = [x_\alpha(t+\Delta t) - x_\alpha(t)] + [x_\alpha(t+2\Delta t) - x_\alpha(t+\Delta t)] \\ &\qquad + \cdots + [x_\alpha(t+\lfloor t_\alpha/\Delta t\rfloor\Delta t) - x_\alpha(t+(\lfloor t_\alpha/\Delta t\rfloor - 1)\Delta t)] \\ &\qquad + [x_\alpha(t+t_\alpha) - x_\alpha(t+[t_\alpha/\Delta t]\Delta t)] \end{aligned} \quad (5)$$

B. COMMUNICATION MODEL

When the task vehicle has a task to calculate, it needs to assign the task to available service vehicles for task execution time reduction. Receiving the assigned task by this task vehicle, service vehicles are responsible for implementing the task. To this end, we adopt the WBSS (WAVE-based basic service set) communication mode after analyzing the

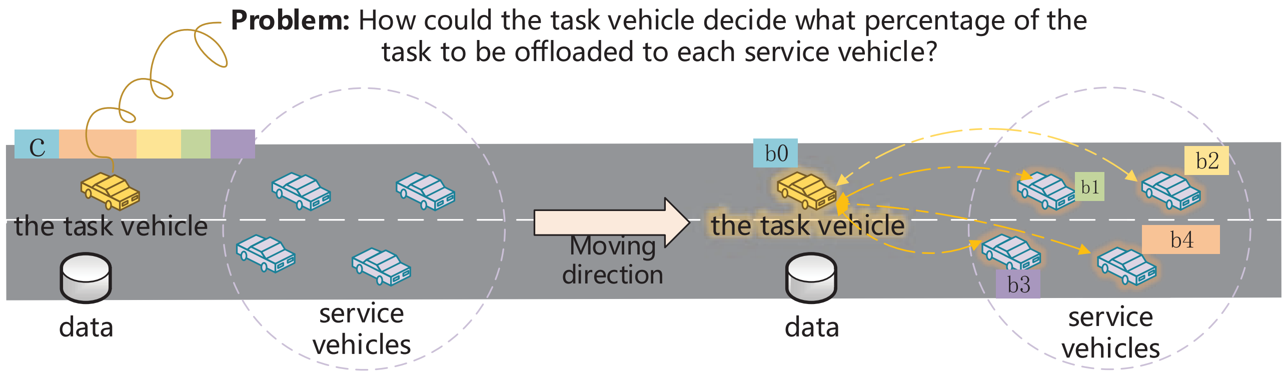

Problem: How could the task vehicle decide what percentage of the task to be offloaded to each service vehicle?
C
the task vehicle
data
service vehicles
Moving direction
b0
the task vehicle
data
b1
b2
b3
b4
service vehicles

FIGURE 2. The illustration of system model, where b_1, b_2, b_3, b_4 indicates the offloaded task proportion.

IEEE 802.11p/WAVE (Wireless Access for Vehicular Environment) standard [37]. The task vehicle broadcasts a WBSS announcement message, and the WAVE Service Advertisement(WSA) on the control channel (CCH), which contain all the information identifying the WAVE application and the network parameters are necessary to join a WBSS [38], [39]. Especially, WAVE application corresponding to the task needs to be processed and these network parameters include the WBSS ID and the service channel (SCH) which this WBSS will use. By monitoring the CCH, service vehicles are capable of knowing about the initiated WBSS by the task vehicle and then join this WBSS by simply switching to the specified SCH. Switching between CCH and SCH adopts immediate channel access [37], which allows immediate communications access on a designated channel for an extended period without consideration for time slot boundaries. We use t_{CCH} to indicate the time spent on the CCH for the task vehicle and service vehicles. t_{SCH} is the transmission delay taken on the SCH for transmitting *data*, which can be written as

$$t_{SCH} = \frac{data}{B\log(1+\frac{P\cdot G}{\sigma^2})} \tag{6}$$

where B is the bandwidth of the SCH that the WBSS initiated by the task vehicle will use. P is the transmission power of the task vehicle. G is the channel gain between the task vehicle and service vehicles. σ^2 is the noise power.

For simplicity, the time for the task vehicle and service vehicles to access the CCH and SCH via the carrier sense multiple access protocol with collision avoidance (CSMA/CA) mechanism is negligible, and the time spent on channel switching, such as guard interval, also is negligible. So the total time for transmitting *data* can be obtained as follows:

$$t^{tr} = t_{CCH} + t_{SCH} \tag{7}$$

Because the calculation result after task processing is quite small [20], the link transmission delay of service vehicles feeding back the calculation result to the task vehicle is neglected.

C. COMPUTATION MODEL

Before introducing the computation model, some assumptions are given:

- **Assumption** 1: the computation task can be divided into two or more parts.
- **Assumption** 2: there are two options for task calculation. All of the computation task can be calculated locally, or part offloaded to multiple service vehicles for processing.

1) THE TASK EXECUTION TIME OF LOCAL COMPUTING

For local task computing, we define f_0 as the CPU computing capability of the task vehicle. Then, the local execution time needed for implementing the task is expressed as follows:

$$t_{local} = c/f_0 \tag{8}$$

2) THE TASK EXECUTION TIME OF OFFLOADING COMPUTING

Denote $b_i \in [0, 1)$ be the task proportion which service vehicle i is responsible for processing. In other words, the task offloading strategy is presented by $b = \{b_1, b_2, b_3 \cdots b_N\}$. Define f_i as the CPU computing capability of service vehicle i. Then, the time needed for service vehicle i to execute its assigned task workload is obtained as follow:

$$t_i^c = \frac{b_i \cdot c}{f_i} \tag{9}$$

Based on the communication model described above, the *data* is broadcasted by the task vehicle. So, the time needed for the service vehicle to receives the *data* from the task vehicle can be obtained as follows:

$$t_i^{tr} = t^{tr} + \delta \tag{10}$$

where δ is the propagation delay.

Thus, the total time for service vehicle i to process the assigned workload by the task vehicle is given as:

$$t_i = t_i^c + t_i^{tr} \tag{11}$$

On the other hand, the needed time for task vehicle to process the rest task workload is written as:

$$t_0 = c(1 - \sum_{i=1}^{N} b_i)/f_0 \tag{12}$$

According to Eq.(11) and Eq.(12), the total task execution time of task vehicle is calculated as:

$$t_{edge} = \max(t_i), \quad i = 0, 1, 2 \cdots N \tag{13}$$

D. PROBLEM FORMULATION

With the aim to minimize the task execution time, the following optimization problem is formulated:

$$\begin{aligned} & minimize\ T = s \cdot t_{local} + (1-s) \cdot t_{edge} \\ & subject\ to\ C1: \sum_{i=1}^{N} b_i < 1 (0 \le b_i < 1) \\ & \qquad C2: t_i \le t_{res\_i} \quad (i = 1, 2 \cdots N) \\ & \qquad C3: |\Delta X_i| \le 2\,|x_0(t) - x_i(t)| \quad (i = 1, 2 \cdots N) \end{aligned} \tag{14}$$

herein, $s = 1$ if the total task is implemented locally, $s = 0$, otherwise. t_{res_i} indicates the residence time of service vehicle i in assembly point. $x_0(t)$ and $x_i(t)$ represent the X-dimensional coordinate of task vehicle and service vehicle i at time t, respectively. ΔX_i indicates the relative displacement between task vehicle and service vehicle i after t_i.

Constraint C1 corresponds to **Assumption** 2. Constraint C2 guarantees that each service vehicle can finish the task computing before leaving the assembly point. Constraint C3 ensures that the link between the task vehicle and its service vehicle keeps available during task offloading. Specially, Constraint C3 is explained as follows: Due to the mobility, the value of ΔX_i is changing with time as shown in Fig.3. Based on Eq.(5), we can get

$$\begin{aligned} \Delta X_i &= \Delta X_i(t_i) - \Delta X_0(t_i) \\ &= [x_i(t + t_i) - x_i(t)] - [x_0(t + t_i) - x_0(t)]. \end{aligned} \tag{15}$$

In order to guarantee the success of task offloading, service vehicle i should not move out of the communicate range of task vehicle with dis as the radius. It is known that a straight line is the shortest distance between two points. Thus, according to the properties of isosceles triangle, the maximum value of ΔX is equal to $2\,|x_0(t) - x_i(t)|$.

III. PROBLEM SOLVING

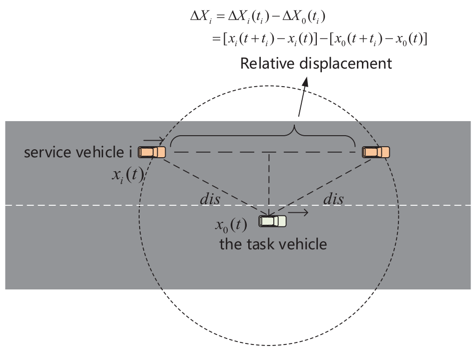
In this section, we present an algorithm to solve the problem in Eq.(14).

A. THE OPTIMAL SOLUTION FOR SPECIAL CASE

In this section, a special case of computation offloading is considered where all service vehicles participate in task processing.

When $s = 1, T = t_{local} = c/f_0$. If $s = 0$, then

$$T = t_{edge} = \min\{\max(t_i)\} \quad i = 0, 1, 2 \cdots N \tag{16}$$



$\Delta X_i = \Delta X_i(t_i) - \Delta X_0(t_i)$
$= [x_i(t+t_i) - x_i(t)] - [x_0(t+t_i) - x_0(t)]$
Relative displacement
service vehicle i
$x_i(t)$
dis
dis
$x_0(t)$
the task vehicle

FIGURE 3. The sketch of Constraint C3.

For the optimal problem in Eq.(16), we adapt Max-Min Fairness Algorithm to solve it. According to Max-Min Fairness Algorithm, by reasonably adjusting the size of b_i, we can guarantee each vehicle (including N service vehicles and the task vehicle) has the same service time, i.e.,

$$t_0 = t_1 = t_2 = \cdots = t_N \tag{17}$$

According Eq.(17), we have:

$$\begin{aligned} b_{equal\_1} \cdot \frac{c}{f_1} + t_1^{tr} &= b_{equal\_2} \cdot \frac{c}{f_2} + t_2^{tr} \\ &= \cdots = b_{equal\_N} \cdot \frac{c}{f_N} + t_N^{tr} \end{aligned} \tag{18}$$

Based on Eq.(18), we can get:

$$b_{equal\_i} = \frac{f_i}{f_1} b_{equal\_1} + \frac{t_1^{tr} - t_i^{tr}}{c} f_i (i = 1, 2, \cdots N) \tag{19}$$

Under the condition that $t_0 = t_1$, the following equation holds:

$$\frac{c}{f_0}(1 - \sum_{i=1}^{N} b_{equal\_i}) = b_{equal\_1} \cdot \frac{c}{f_1} + t_1^{tr} \tag{20}$$

where b_{equal_i} is the task segmentation ratio of each service vehicle under the condition of average distribution.

By combining Eq.(19) and Eq.(20), we have:

$$b_{equal\_1} = \frac{c/f_0 - t_1^{tr} - \sum_{i=1}^{N} \frac{f_i}{f_0}(t_1^{tr} - t_i^{tr})}{\frac{c}{f_1} + \frac{c}{f_0} \cdot \sum_{i=1}^{N} \frac{f_i}{f_1}} \tag{21}$$

As a result, the service time of each service vehicle is expressed as:

$$t_0 = t_1 = t_2 = \cdots = t_N = T_{equal} = b_{equal\_1} \frac{c}{f_1} + t_1^{tr} \tag{22}$$

However, due to the mobility, the link between the task vehicle and its service vehicle may be not available for such a long time. For Constraint C3, according to Eq.(4) and Eq.(5), we can see that the displacement and time are quadratic inequality relations, which are solvable. We assume that the solution is $[0, t_{constraint\_c3}]$. Then, combining Constraint C2, we can get $0 < t_i \le t_{\max\_i}$

($t_{\max\_i} := \min\{t_{res\_i}, t_{constraint\_c3}\}$). By ascending sort, $t_{\max\_1} \leq t_{\max\_2} \leq t_{\max\_3} \leq \cdots \leq t_{\max\_N}$. When compared with T_{equal}, $t_{\max\_l} < T_{equal} \leq t_{\max\_r}$ $l, r \in \{1, 2, \cdots, N\}$ and $l < r$. So, the task workloads assigned to $Vidle_1$, $Vidle_2$, $\cdots$, $Vidle_l$ are too big and need to be further adjusted.

First, it is required to calculate all extra task offloading ratios. Take $Vidle_1$ as an example. The maximum task offloading ratio that $Vidle_1$ can undertake is $b_{\max\_1}$, which is obtained as follows:

$$\begin{aligned} t_{\max\_1} &= b_{\max\_1} \cdot \frac{c}{f_1} + t_1^{tr} \\ \Rightarrow b_{\max\_1} &= \frac{t_{\max\_1} - t_1^{tr}}{c} f_1 \end{aligned} \tag{23}$$

As a result, the extra task offloading ratio of $Vidle_1$ is calculated as:

$$b_{extra\_1} = b_{equal\_1} - b_{\max\_1} \tag{24}$$

Correspondingly, all extra task offloading ratios can be given by:

$$b_{extra} = \sum_{i=1}^{l} b_{extra\_i} \tag{25}$$

Then, the extra task offloading ratio b_{extra} will be distributed to $Vidle_{l+1}, Vidle_{l+2}, Vidle_{l+3}, \cdots, Vidle_N$ and the task vehicle. Similar to the above uniform distribution, the following equation holds:

$$\begin{aligned} b_{equal2\_i} &= \frac{f_i}{f_{l+1}} b_{equal2\_l+1} \\ &\quad + \frac{t^{tr}_{equal2\_l+1} - t_i^{tr}}{c} f_i (i = l+2, \cdots N) \\ b_{equal2\_l+1} \cdot \frac{c}{f_{l+1}} + t^{tr}_{l+1} &\\ = \frac{c}{f_0}(b_{extra} - \sum_{i=l+1}^{N} b_{equal2\_i}) &\\ b_{equal2\_l+1} = \frac{b_{extra} \cdot c/f_0 - t^{tr}_{l+1} - \sum\limits_{i=l+1}^{N} \frac{f_i}{f_0}(t^{tr}_{l+1} - t^{tr}_i)}{\frac{c}{f_{l+1}} + \frac{c}{f_0} \cdot \sum\limits_{i=l+1}^{N} \frac{f_i}{f_{l+1}}} & \end{aligned} \tag{26}$$

where b_{equal2_i} is the task workload assigned to $Vidle_i$, $i = l+1, l+2, \cdots, N$ from b_{extra}. The task offloading ratio assigned to $Vidl_i$ is: $b_i = b_{equal\_i} + b_{equal2\_i}, i = l+1, l+2, \cdots N$. Similarly, we should check whether $Vidl_i$ can undertake such big task offloading ratio according to its maximum service time. If no, we need to follow the above method to assign extra segmentation ratio. Repeat the assignment scheme again and again until all service vehicles are not overloaded. After that, We can get the final $b_i, i = 1, 2, 3, \cdots N$, and $T = t_{edge} = \max(t_i) i = 0, 1, 2 \cdots N$.

Finally, we compare the value of t_{local} and t_{edge}. If $t_{local} \leq t_{edge}$, it is better to adapt local computing ($s = 1$), otherwise, we should offload the task to service vehicles based on the final b_i.

B. THE FEASIBLE SOLUTION FOR GENERAL CASE

In this section, a more general case is considered where we do not know whether each service vehicle will participate in task processing. In order to solve the problem in Eq.(16), 2^N program is needed, which brings difficulty to the analysis and optimization of the optimal problem. With the aim to deal with this issue, we employ the PSO Algorithm to solve the problem in Eq.(16).

According to the PSO Algorithm, define the position of the j^{th} particle in the N-dimensional space at step k as $B_j^k = (b_{j1}^k, b_{j2}^k, \cdots, b_{jN}^k)^T, j = 1, 2, \cdots, l$, the velocity as $V_j^k = (v_{j1}^k, v_{j2}^k, \cdots, v_{jN}^k)^T, j = 1, 2, \cdots, l$. The best position that j^{th} particle has experienced is $pbest_j^k = (p_{j1}^k, p_{j2}^k, \cdots, p_{jN}^k)^T, j = 1, 2, \cdots, l$. The best position that the particle group has experienced is $gbest^k = (g_1^k, g_2^k, \cdots, g_N^k)^T$. The fitness function is shown in Eq.(27):

$$T_j^k = t_{edge}(B_j^k) \quad (j = 1, 2, \cdots, l) \tag{27}$$

What is more, when updating the best position of each particle, we not only take into account the fitness function, but also take into account the number of service vehicles involved in the calculation. That is, when the fitness function is reduced, the number of service vehicles participating in the calculation must also be reduced.

The j^{th} particle updates its velocity and position according to the following formula:

$$\begin{aligned} V_j^{k+1} &= wV_j^k + c_1 r_1 (pbest_j^k - B_j^k) + c_2 r_2 (gbest^k - B_j^k) \\ B_j^{k+1} &= B_j^k + V_j^{k+1} \end{aligned} \tag{28}$$

where w is inertia weight, c_1 and c_2 are learning weight, the elements of vector r_1 and vector r_2 are assigned values from a uniform random distribution between -1 and 1 which are independent with each other.

However, we need to be aware that in the process of initialization and iteration according to Eq.(28), the position B_j^k of each particle must meet Constraint C1-C3. So, we make the corresponding adjustment to the PSO Algorithm. When we find that the current particle's location information does not meet the constraints, we will adjust according to the following steps:

Step1. For Constraint C1, we count the sum of the N-dimensional positions of the current particles. If the sum is greater than 1, for the part beyond 1, the position of each dimension is reduced according to its proportion. Taking the 1-dimensional position of j^{th} particle b_{j1}^{k+1} as an example during the k-th iteration based on the above adjustment rules, we can get the following equation:

$$b_{j1}^{k+1} = b_{j1}^k - \frac{b_{j1}^k}{\sum\limits_{i=1}^{N} b_{ji}^k} (\sum_{i=1}^{N} b_{ji}^k - 1) \tag{29}$$

Step2. For Constraint C2 and Constraint C3, based on the result of the **Step1** adjustment, we check each dimension position of the current particle in turn. If its value is greater than its maximum value according Eq.(23), directly let its value be equal the maximum value. The PSO Algorithm implementation steps are shown in **Algorithm 1**.

Algorithm 1 PSO Algorithm to Solve the Problem in Equation(16)

```
Input:
    The parameter of the task vehicle, $f_0, X_0(t)$;
    The parameter of all service vehicles, $f_i, X_i(t)$;
    The parameter of the task, $\{c, data, t^{\max}\}$;
Output:
    Get $b_i(gbest)$;
 1: Set $k = 1$, iterative number K, error criterion $\delta$, $t_{edge}(gbest^0) = \infty$;
 2: According to Constraint C1-C3, randomly determine the initial position and velocity of $l$ particles;
 3: Check and adjust the initial position of $l$ particles;
 4: while $k \le K || t_{edge}(gbest^k) - t_{edge}(gbest^{k-1}) \ge \delta$ do
 5:     for $j = 1, 2, \cdots, l$ do
 6:         calculate fitness function $T_j^k = t_{edge}(B_j^k)$, update $pbest_j^k$;
 7:     end for
 8:     Update $gbest^k$
 9:     $k = k + 1$
10:     for $j = 1, 2, \cdots, l$ do
11:         Update position and velocity of $l$ particles according Eq.(28);
12:     end for
13: end while
14: return $gbest^K$;
```

Through PSO Algorithm, we can get the value of b_i and $t_{edge} = \max(t_i)$. Similar to the processing method in the Special Case, we compare t_{local} and t_{edge}, and then decide whether to offload the task to service vehicles.

IV. PERFORMANCE EVALUATION

In this section, we carry out simulations to evaluate the performance of the proposed scheme. In order to simulate realistic traffic, we use the TIGER map provided by The United States Census Bureau, and then adopt the IDM_IM model provided by VanetMobiSim [34] to generate vehicle track files. Then, we analyze the track files to obtain the moving parameters of vehicles in the resource pool, such as $t_{\max\_i}$, and use ns3 to obtain related parameters of the IEEE 802.11p/WAVE standard, such as reliable communication distance *dis*. Based on the parameters above, we compare the performance of three computing schemes: Local computing scheme, Offloading scheme with Max-Min Fairness Algorithm and Offloading scheme with PSO Algorithm. The used main simulation parameters are listed in Table 2.

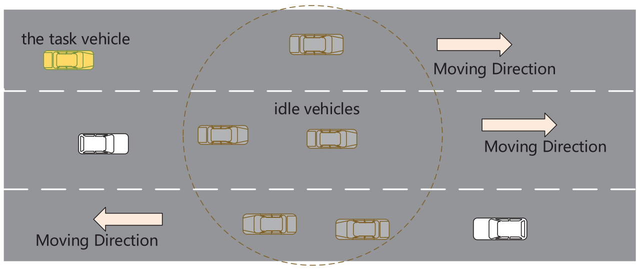


FIGURE 4. The scene of offloading computing.

TABLE 2. Summary of the simulation parameters.

Parameter	Value
Simulation area	$2km \times 2km$
Mobility model	IDM_IM model
Communication range dis	115m
The amount of computing resource required for the task c	Uniform in $[2 \times 10^9, 30 \times 10^9]$
Data size of computation task $data$ (KB)	Uniform in [300, 1200]
The completion deadline for the task $t^{\max}$ (s)	Uniform in [0.5, 5]
The computing capabilities of the task vehicle and multi service vehicles f_i (GHz)	Uniform in [0.2, 1]
Total number of service vehicles N	From 3 to 30

First, we compare the performance of three schemes over different number of service vehicles. The task execution time is greatly reduced in Fig.5(a) by the Max-Min Fairness Algorithm and the PSO Algorithm, respectively. Because we unload the part of the task to the service vehicles, and use the idle computing resources of the service vehicle to help calculate the task. From Fig.5(a), we can see that with the increase of the number of service vehicles, the task execution time in the latter two schemes is slowly decreasing. Because the more service vehicles, the more idle computing resources that can be scheduled for the task vehicle. The task vehicle can unload a larger proportion of the task to the service vehicle. Not only that, from Fig.5(a), at the beginning, by the Max-Min Fairness Algorithm and the PSO Algorithm, the task execution time is reduced to the same extent. But when the number of service vehicles reaches 7, the performance using the PSO Algorithm starts to be slightly lower than that using the Max-Min Fairness Algorithm. However, it can be seen from Fig.5(b) that the PSO algorithm does not call all service vehicles at the same time. The cost of calling an service vehicle is not considered in this paper. If the service vehicle is called for a fee, the PSO Algorithm will greatly reduce the cost of the unloading calculation. On the other hand, this paper does not consider the scenario of multiple tasks. If there are multiple tasks in the future that need to be unloaded, we can also get better performance by using PSO Algorithm. Because each task only needs to be unloaded to a part of service vehicles. The remaining service vehicles

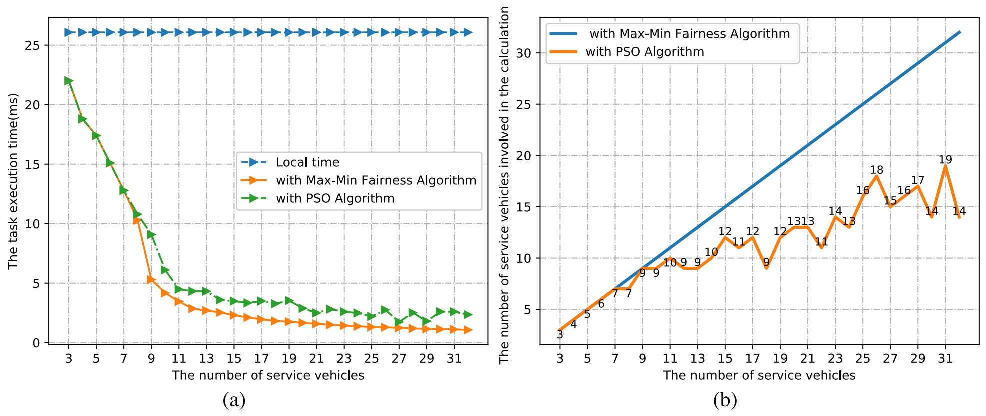


FIGURE 5. Effect of three schemes. (a)The task execution time over different number of service vehicles; (b)The number of service vehicles involved in the calculation over different number of service vehicles. The default setting is $c = 20 \times 10^9$, $data = 750KB$, $t^{max} = 2.5$, $l = 150$, $K = 200$. The value of f_i is set according to Table 2. The number marked on the curve indicates the number of service vehicles.

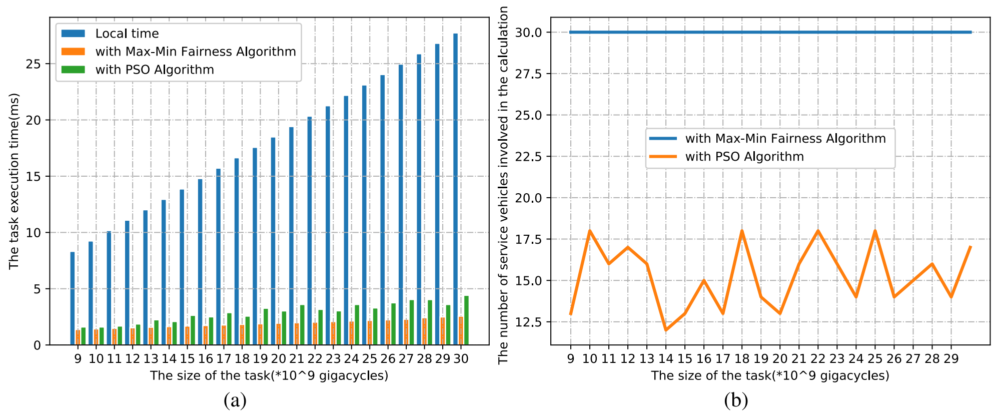


FIGURE 6. Effect of three schemes. (a)The task execution time over different size of the task; (b)The number of service vehicles involved in the calculation over different size of the task. The default setting is $N = 30$, $data = 750KB$, $t^{max} = 2.5$, $l = 150$, $K = 200$. The value of f_i is set according to Table 2.

can be scheduled for other tasks. Compared with the Max-Min Fairness Algorithm, the PSO Algorithm does not call all service vehicles at the cost of not minimizing task execution time. Considering the performance of the task execution time and the number of service vehicles involved in the calculation, we can see that the PSO Algorithm is slightly better than the Max-Min Fairness Algorithm.

Second, we compare the robustness of three schemes over different size of the task. In order to better compare the performance of the three schemes, the task size we set is relatively large at the beginning. When the task size grows linearly, the task execution time locally also increases linearly in Fig.6(a). The reason is that only the computing resources of the task vehicle can be used in this case. The robust performance of this scheme is the worst. For remaining two schemes, although the task execution time increases with the increase of the task size, the growth rate is much lower than the task execution time locally. In addition, the performance using the PSO Algorithm is also still slightly lower than using the Max-Min Fairness Algorithm. Similarly, from Fig.6(b), we also find that all service vehicles are always scheduled using the Max-Min Fairness Algorithm. With the PSO Algorithm, the number of service vehicles scheduled fluctuates with the increasing the task size. But the fluctuation range is small. And in the worst case, 18 service vehicles are called. In the best case, less than half of the service vehicles are called. All the results show the third scheme using the PSO Algorithm has the best robust performance.

Next, we analyze the impact of relative movement between vehicles on the performance of these three schemes. According to the IDM_IM mobility model we use, the traffic light time, the initial position and initial velocity of the vehicles will affect the relative displacement ΔX_i between the task vehicle and each idle vehicle. Based on Constraint 3, the communication time $t_{constraint\_c3}$ between them will be affected, further affects the value of $t_{\max\_i}$. In the simulation process, we get different $t_{constraint\_c3}$ by setting different movement parameters in the VanetMobiSim software, and then get different $t_{\max\_i}$. But we still set the abscissa of Fig.7 directly to different $t_{\max\_i}$ instead of different movement parameters. From Fig.7(a), we can see that with the increase of $t_{\max\_i}$,

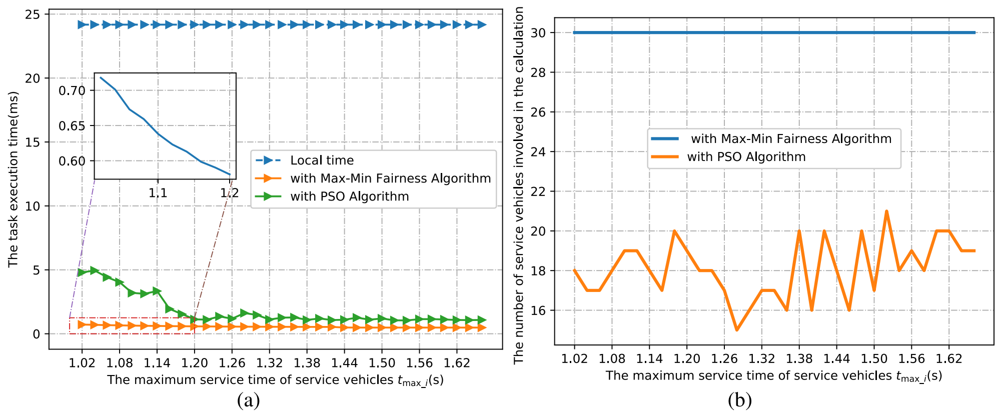


FIGURE 7. Effect of three schemes. (a)The task execution time over different service time of service vehicles; (b)The number of service vehicles involved in the calculation over different service time of service vehicles. The default setting is $c = 10 \times 10^9$, $data = 750KB$, $t^{\max} = 2.5$, $l = 150$, $K = 200$. The value of f_j is set according to Table 2.

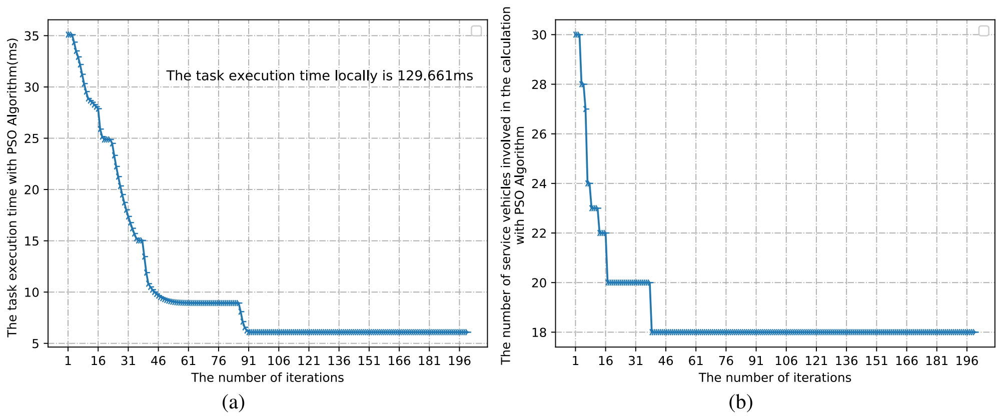


FIGURE 8. Effect of the PSO Algorithm. (a)The task execution time over different number of iterations; (b)The number of service vehicles involved in the calculation over different number of iterations. The default setting is $N = 30$, $data = 750KB$, $t^{\max} = 2.5$, $l = 150$. The value of f_j is set according to Table 2. The number marked on the curve indicates the coordinates.

the task execution time in the scheme using the Max-Min Fairness Algorithm and the PSO Algorithm keeps decreasing until it is stable. Because the value of $t_{\max\_i}$ is greater, the task vehicle could allocate a larger proportion of the task to the service vehicles for reducing the task execution time. From Fig.7(b), the number of service vehicles called by the PSO Algorithm fluctuates around 18, and nearly half of the service vehicles are not scheduled.

Finally, we analyze the performance of the scheme using the PSO Algorithm. In Fig.8(a), when the number of iterations reaches 91, the task execution time is reduced to a minimum value, which is much smaller than the task execution time locally. This shows that using the PSO Algorithm does not require iterating many times to achieve optimal performance. The task vehicle does not need to spend too much time cost to schedule service vehicles with the PSO Algorithm. Through Fig.8(b), we can see that as the number of iterations increases, the PSO Algorithm optimizes the task execution time while the number of service vehicles scheduled is also optimal. This shows the PSO Algorithm can obtain good performance in the iterative process.

V. CONCLUSION

In this paper, to reduce the expensive charge from cellular or infrastructure communication, and guarantee the desired latency for delay-sensitive applications in IoV, a task offloading scheme merely relying on V2V communication is proposed. By formulating the one task and multiple cooperative offloading pattern into a Min-Max problem, the total task execution time can be optimized with affordable communication and computing overhead. Numerical results have demonstrated that our scheme could outperform some state-of-the-art strategies in terms of task execration time and consumed resources. In addition, the introduced PSO algorithm also contributes a lot to the convergence speed and accuracy of our envisioned problem. Our future work will investigate the possibility to enroll multiple tasks into the offloading procedure using V2V communication, and analyze the impact of high mobility of vehicles on the offloading performance.

REFERENCES

[1] H. Cao, S. Wu, G. S. Aujla, Q. Wang, L. Yang, and H. Zhu, "Dynamic embedding and quality of service driven adjustment for cloud networks," *IEEE Trans. Ind. Informat.*, to be published, doi: 10.1109/tii.2019.2936074.

[2] S. Mao, S. Leng, J. Hu, and K. Yang, "Energy-efficient transmission schemes for cooperative wireless powered cellular networks," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 2, pp. 494–504, Jun. 2019.
[3] H. Cao, Y. Zhu, G. Zheng, and L. Yang, "A novel optimal mapping algorithm with less computational complexity for virtual network embedding," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 356–371, Mar. 2018.
[4] S. He, W. Tian, J. Zhang, K. Li, M. Zhang, and R. Zhu, "A high efficient approach for power disturbance waveform compression in the view of heisenberg uncertainty," *IEEE Trans. Ind. Inf.*, vol. 15, no. 5, pp. 2580–2591, May 2019.
[5] X. Yuan, C. Li, Q. Ye, K. Zhang, N. Cheng, N. Zhang, and X. Shen, "Performance analysis of IEEE 802.15.6-based coexisting mobile WBANs with prioritized traffic and dynamic interference," *IEEE Trans. Wireless Commun.*, vol. 17, no. 8, pp. 5637–5652, Aug. 2018.
[6] Y. Dai, D. Xu, S. Maharjan, G. Qiao, and Y. Zhang, "Artificial intelligence empowered edge computing and caching for Internet of vehicles," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 12–18, Jun. 2019.
[7] M. M. K. Tareq, O. Semiari, M. A. Salehi, and W. Saad, "Ultra reliable, low latency vehicle-to-infrastructure wireless communications with edge computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–7.
[8] F. Yang, J. Li, T. Lei, and S. Wang, "Architecture and key technologies for Internet of vehicles: A survey," *J. Commun. Inf. Netw.*, vol. 2, no. 2, pp. 1–17, Jun. 2017.
[9] C. Chen, Y. Jin, Q. Pei, and N. Zhang, "A connectivity-aware intersection-based routing in VANETs," *EURASIP J. Wireless Commun. Netw.*, vol. 2014, no. 1, p. 42, 2014.
[10] C. Chen, Q. Pei, and X. Li, "A GTS allocation scheme to improve multiple-access performance in vehicular sensor networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 3, pp. 1549–1563, Mar. 2016.
[11] J. Guan, R. Lai, and A. Xiong, "Wavelet deep neural network for stripe noise removal," *IEEE Access*, vol. 7, pp. 44544–44554, 2019.
[12] Y. Yang, D. Li, and Z. Duan, "Chinese vehicle license plate recognition using kernel-based extreme learning machine with deep convolutional features," *IET Intell. Transp. Syst.*, vol. 12, no. 3, pp. 213–219, Apr. 2018.
[13] Y. Dai, D. Xu, S. Maharjan, Z. Chen, Q. He, and Y. Zhang, "Blockchain and deep reinforcement learning empowered intelligent 5G beyond," *IEEE Netw.*, vol. 33, no. 3, pp. 10–17, May 2019.
[14] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[15] S. Xiao-Hong, "Key technology and its application of IoV," *Commun. J.*, vol. 4, pp. 47–50, 2013.
[16] X. Xu, Y. Xue, L. Qi, Y. Yuan, X. Zhang, T. Umer, and S. Wan, "An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles," *Future Gener. Comput. Syst.*, vol. 96, pp. 89–100, Jul. 2019.
[17] Z. Duan, Y. Yang, K. Zhang, Y. Ni, and S. Bajgain, "Improved deep hybrid networks for urban traffic flow prediction using trajectory data," *IEEE Access*, vol. 6, pp. 31820–31827, 2018.
[18] S. Mao, S. Leng, J. Hu, and K. Yang, "Power minimization resource allocation for underlay MISO-NOMA SWIPT systems," *IEEE Access*, vol. 7, pp. 17247–17255, 2019.
[19] P. Guo, W. Hou, L. Guo, W. Sun, C. Liu, H. Bao, L. H. K. Duong, and W. Liu, "Fault-tolerant routing mechanism in 3D optical network-on-chip based on node reuse," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 3, pp. 547–564, Mar. 2020, doi: 10.1109/tpds.2019.2939240.
[20] C. Li, S. Wang, X. Huang, X. Li, R. Yu, and F. Zhao, "Parked vehicular computing for energy-efficient Internet of vehicles: A contract theoretic approach," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6079–6088, Aug. 2019, doi: 10.1109/jiot.2018.2869892.
[21] S. Mao, S. Leng, K. Yang, Q. Zhao, and M. Liu, "Energy efficiency and delay tradeoff in multi-user wireless powered mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2017, pp. 1–6.
[22] J. Guo, B. Song, F. R. Yu, Y. Chi, and C. Yuen, "Fast video frame correlation analysis for vehicular networks by using CVS-CNN," *IEEE Trans. Veh. Technol.*, vol. 68, no. 7, pp. 6286–6292, Jul. 2019.
[23] J. Guo, B. Song, Y. He, F. R. Yu, and M. Sookhak, "A survey on compressed sensing in vehicular infotainment systems," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2662–2680, 4th Quart., 2017.
[24] Y. Wang, Y. Zhang, M. Sheng, and K. Guo, "On the interaction of video caching and retrieving in multi-server mobile-edge computing systems," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1444–1447, Oct. 2019.
[25] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and C.-L. I, "Enabling low-latency applications in LTE—A based mixed fog/cloud computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1757–1771, Feb. 2019.
[26] J. Luo, X. Deng, H. Zhang, and H. Qi, "Ultra-low latency service provision in edge computing," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[27] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.
[28] 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, no. 6, pp. 3860–3873, Jun. 2016.
[29] S. Li, Q. Ni, Y. Sun, G. Min, and S. Al-Rubaye, "Energy-efficient resource allocation for industrial cyber-physical IoT systems in 5G era," *IEEE Trans. Ind. Informat.*, vol. 14, no. 6, pp. 2618–2628, Jun. 2018.
[30] J. Zhou, Y. Sun, Q. Cao, S. Li, H. Xu, and W. Shi, "QoS-based robust power optimization for SWIPT NOMA system with statistical CSI," *IEEE Trans. Green Commun. Netw.*, vol. 3, no. 3, pp. 765–773, Sep. 2019, doi: 10.1109/tgcn.2019.2914736.
[31] L. Liu, C. Chen, Q. Pei, S. Maharjan, and Y. Zhang, "Vehicular edge computing and networking: A survey," 2019, *arXiv:1908.06849*. [Online]. Available: https://arxiv.org/abs/1908.06849
[32] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.
[33] N. Ababneh and H. Labiod, "A performance analysis of VANETs routing protocols using different mobility models," in *Proc. IEEE Int. Conf. Wireless Commun., Netw. Inf. Secur. (WCNIS)*, Jun. 2010, pp. 498–502.
[34] J. Härri, F. Filali, C. Bonnet, and M. Fiore, "VanetMobiSim: Generating realistic mobility patterns for VANETs," in *Proc. 3rd Int. Workshop Veh. Ad Hoc Netw.*, 2006, pp. 96–97.
[35] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 62, no. 2, p. 1805, 2000.
[36] M. Treiber and D. Helbing, "Explanation of observed features of self-organization in traffic flow," 1999, *arXiv:cond-mat/9901239*. [Online]. Available: https://arxiv.org/abs/cond-mat/9901239
[37] *IEEE Standard for Wireless Access in Vehicular Environments (WAVE)—Multi-Channel Operation*, IEEE Standard 1609.4-2016, 2016, pp. 4–1609.
[38] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing IEEE 802.11 p/WAVE to provide infotainment applications in VANETs," *Ad Hoc Netw.*, vol. 10, no. 2, pp. 253–269, 2012.
[39] C. Campolo and A. Molinaro, "Data rate selection in WBSS-based IEEE 802.11 p/WAVE vehicular ad hoc networks," in *Proc. 7th Int. Symp. Commun. Syst., Netw. Digit. Signal Process. (CSNDSP)*, 2010, pp. 412–416.

CHEN CHEN (Senior Member, IEEE) received the B.Eng., M.Sc., and Ph.D. degrees in telecommunication from Xidian University, Xi'an, China, in 2000, 2006, and 2008, respectively. He was a Visiting Professor with the EECS in the University of Tennessee and the CS in the University of California. He is currently an Associate Professor with the Department of Telecommunication, Xidian University. He is also the Director of the Xi'an Key Laboratory of Mobile Edge Computing and Security, and the Director of the Intelligent Transportation Research Laboratory, Xidian University. He has authored/coauthored three books, over 90 scientific articles in international journals and conference proceedings. He has contributed to the development of five copyrighted software systems and invented 90 patents. He is also a member of ACM, Chinese Institute of Electronics and a Senior Member of China Computer Federation (CCF). He also serves as a General Chair, a PC Chair, and a Workshop Chair or a TPC Member of a number of conferences.

LANLAN CHEN received the B.Eng. degree in communication engineering from the Heifei Univeesity of Techology, Heifei, China, in 2013. She is currently pursuing the master's degree in transportation information engineering and control with Xi'an University. Her research interests include vehicular ad hoc networks, named data networks, mobile edge computing, and the Internet of Thing.

LEI LIU (Student Member, IEEE) received the B.Eng. degree in communication engineering from Zhengzhou University, Zhengzhou, China, in 2010, and the M.Sc. degree in communication engineering from Xidian University, Xi'an, China, in 2013. He is currently pursuing the Ph.D. degree with Xidian University. He has published more than ten journals and conference publications. His research interests include vehicular ad hoc networks, intelligent transportation, mobile edge computing, and the Internet of Thing.

SHUNFAN HE was born in Wuhan, Hubei, China, in 1984. He received the Ph.D. degree in power electrical engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2013. He is currently with the College of Computer Science, South-Central University for Nationalities, Wuhan, China. He has authored more than ten articles in international journals in the areas of electromagnetic compatibility, big data processing, electronic transducers, and power disturbance analysis. His current research interests include electromagnetic compatibility, big data processing, and power disturbance analysis.

XIAOMING YUAN received the B.E. degree in electronics and information engineering from Henan Polytechnic University, China, in 2012, and the Ph.D. degree in communication and information system from Xidian University, China, in 2018. From 2016 to 2017, she was a Visiting Scholar with the Broadband Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. She is currently with Northeastern University at Qinhuangdao. Her research interests include cloud/edge computing, medium access control and performance analysis for wireless body area networks, and the Internet of Things.

DAPENG LAN received the B.Eng. degree in microelectronics from Sun Yat-sen University, Guangzhou, China, in 2014, the M.Sc. degree in ICT innovation from the KTH Royal Institute of Technology, Stockholm, Sweden, in 2016, and the M.Sc. degree in innovation in information and communication technology from the Technical University of Berlin, Berlin, Germany, in 2017. In 2017, he was with InnoEnergy, where he was involved in smart building energy management in Sweden. He was a thesis Student with the ABB Corporate Research Center, Västerås, Sweden, in 2016. He is currently a Ph.D. Research Fellow with the Department of Informatics, University of Oslo, Norway. His research interests include fog computing, the Internet of Things, and distributed systems.

ZHUANG CHEN received the B.S. degree in Internet of Things engineering from the Qingdao University of Science and Technology, China, in 2017. He is currently pursuing the M.S. degree with the School of Computer and Information Security, Guilin University of Electronic Technology, China. He is also a Visiting Student with the Department of Informatics, University of Oslo. His current research interests include mobile edge computing, multimedia cache, wireless networks, deep reinforcement learning, and blockchain.

• • •