

Cooperative Task Scheduling for Computation Offloading in Vehicular Cloud

Fei Sun , *Student Member, IEEE*, Fen Hou , *Member, IEEE*, Nan Cheng , *Member, IEEE*,
Miao Wang, *Member, IEEE*, Haibo Zhou, *Senior Member, IEEE*, Lin Gui, *Member, IEEE*,
and Xuemin Shen, *Fellow, IEEE*

Abstract—Technological evolutions in the automobile industry, especially the development of connected and autonomous vehicles, have granted vehicles more computing, storage, and sensing resources. The necessity of efficient utilization of these resources leads to the vision of vehicular cloud computing (VCC), which can offload the computing tasks from the edge or remote cloud to enhance the overall efficiency. In this paper, we study the problem of computation offloading through the vehicular cloud (VC), where computing missions from edge cloud can be offloaded and executed cooperatively by vehicles in VC. Specifically, computing missions are further divided into computing tasks with interdependency and executed in different vehicles in the VC to minimize the overall response time. To characterize the instability of computing resources resulting from the high vehicular mobility, a mobility model focusing on vehicular dwell time is utilized. Considering the heterogeneity of vehicular computing capabilities and the interdependency of computing tasks, we formulate an optimization problem for task scheduling, which is NP-hard. For low complexity, a modified genetic algorithm based scheduling scheme is designed where integer coding is used rather than binary coding, and relatives are defined and employed to avoid infeasible solutions. In addition, a task load based stability analysis of the VCC system is presented for the cases where some vehicles within the VC are offline. Numerical results demonstrate that the proposed scheme can significantly improve the utilization of computing resources while guaranteeing low latency and system stability.

Index Terms—Vehicular cloud, computation offloading, interdependency, heterogeneity, and modified genetic algorithm.

I. INTRODUCTION

A. Motivation and Goal

TECHNOLOGICAL development of mobile devices, such as smartphones and laptops, has motivated the evolution of mobile applications, some of which are computation-intensive, such as augmented reality (AR)/virtual reality (VR), online gaming, face recognition, etc [1]. Due to the limited battery capacity and computing capabilities of users' devices, running these computation-intensive applications locally could lead to excessive energy consumption. One possible way is to offload the computing tasks to the remote centralized cloud (CC), which can utilize the abundant computing resources in the cloud servers to reduce cost and save energy for end devices. However, the remote execution could lead to higher latency due to the long distance from mobile devices to the Internet cloud and pose challenges to the backhaul bandwidth. To cope with these problems, a novel concept named mobile edge computing (MEC) has been proposed, which brings computing and storage resources to the edge of the mobile network to meet the strict delay and bandwidth requirements. Although edge cloud (EC) is beneficial in terms of low delay [2] and less backhaul bandwidth consumption, its computing capability is relatively limited compared to CC [3].

With the development of mobile Internet, intelligent transportation system, smart city, and connected and autonomous vehicles (CAVs), many computation-intensive applications are emerging, such as self-driving, crowdsensing, VR gaming, etc [4]–[7]. These applications require massive computing resources, which could pose severe challenges to the EC. Recently, the evolving CAVs are typically outfitted with significant communication and computing capabilities. For example, the NVIDIA DRIVE PX platform integrates the computation capabilities of deep learning, sensor fusion, and surround vision for autonomous driving and can support up to 320 trillion deep learning operations in one second [8]. Employing the vehicular networks (VANETs) [9], [10], the vehicles with powerful computing capabilities can compose a novel cloud, which is referred to as vehicular cloud (VC) [11]. Through vehicular cloud computing (VCC), not only the vehicle-related

Manuscript received May 2, 2018; revised July 27, 2018; accepted August 22, 2018. Date of publication August 30, 2018; date of current version November 12, 2018. This work was supported in part by the National Natural Science Foundation of China under Grants 61420106008, 61671295, 61471236, and 61871211, in part by the 111 Project under Grant B07022, in part by the National Key Laboratory of Science and Technology on Communications under Grant KX172600030, in part by the Shanghai Key Laboratory of Digital Media Processing and Transmissions, in part by the Macau Science and Technology Development Fund under Grant FDCT 121/2014/A3, in part by the Ministry of Science and Technology of China and Macau Science and Technology Development under Grant 037/2017/AMJ, in part by the China Scholarship Council, and in part by the Natural Sciences and Engineering Research Council, Canada. The review of this paper was coordinated by Prof. L. Guo. (*Corresponding author: Lin Gui.*)

F. Sun and L. Gui are with the Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: sf19912010@sjtu.edu.cn; guilin@sjtu.edu.cn).

F. Hou is with the Department of Electrical and Computer Engineering, University of Macau, Macau 999078, China (e-mail: fenhoul@umac.mo).

N. Cheng and X. Shen are with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L3G1 Canada (e-mail: n5cheng@uwaterloo.ca; sshen@uwaterloo.ca).

M. Wang is with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH 45056 USA (e-mail: wangm64@miamioh.edu).

H. Zhou is with the School of Electronic Science and Engineering, Nanjing University, Nanjing 210093, China (e-mail: haibozhou@nju.edu.cn).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TVT.2018.2868013

computation-intensive tasks can be fulfilled efficiently, but also the capacity of EC can be enhanced by offloading computing tasks to the VC.

Different from the centralized cloud or edge cloud, VC has its unique features, which pose challenges to offloading decision making in VC. Due to the high mobility of vehicles, the computing resources are highly dynamic [12], [13]. For example, a moving vehicle may join or leave the VC at any time, which leads to the instability in resource availability. Therefore, to ensure the completion of a task, it should be scheduled within the contact duration of the vehicles in VC. Actually, this instability is the major difference between VC and CC/EC. Moreover, the inter-dependency of computing tasks will also affect the assignment of the computing resources. Normally, these tasks can be either independent of each other or mutually dependent. In the former case, all tasks can be offloaded simultaneously and processed in parallel. However, in the latter case, some tasks need the output of other tasks as their input data, and thus parallel processing may not be feasible. Therefore, a careful scheduling scheme is important to efficiently offload the computing tasks in VC, considering such features.

In this paper, we consider a VC-assisted computation offloading scenario, where computing missions from EC can be offloaded and processed cooperatively by vehicles in VC, in order to enhance the overall capacity of EC. To guarantee the efficient utilization of the vehicular onboard computing resources, we propose a cooperative task scheduling scheme, aiming at minimizing the average response time of the offloaded missions. In specific, to characterize the instability caused by the mobility of vehicles, a dwell time oriented vehicular mobility model is utilized. Considering the heterogeneity of vehicular computing capabilities and the inter-dependency of computing tasks, we formulate a task scheduling problem, which is NP-hard. A low-complexity modified genetic algorithm (MGA) based scheduling scheme is proposed where integer coding is used, and relatives are defined and employed to avoid infeasible solutions. Additionally, to cope with the possible offline of vehicles, we present a task load based stability analysis of the VCC system to further improve the performance of the MGA based scheduling scheme.

B. Main Contributions

We propose an efficient task scheduling scheme in VC by jointly considering the instability of resources, the heterogeneity of vehicular computing capabilities, and the inter-dependency of computing tasks. Our main contributions are summarized as follows.

- 1) We propose a cooperative task scheduling scheme for computation offloading in VC, which is formulated as an NP-hard scheduling problem, considering the unique features of VC such as instability, heterogeneity, and inter-dependency of computing tasks.
- 2) To characterize the instability of the onboard computing resources, a vehicular mobility model focusing on dwell time is developed, where the dwell time can be directly used in offloading decision making.

- 3) We design and implement a genetic-based heuristic algorithm to solve the stated NP-hard scheduling problem, which reduces the complexity of the problem while enhancing the utilization of the vehicular onboard resources.

C. Related Works

Since vehicular cloud computing is envisioned as a promising approach to enhance the overall capacity of EC and CC, a considerable amount of research has been conducted on the vehicular cloud architecture [14]–[16]. A novel 3-tier VC network architecture is proposed in [14], which consists of the vehicular cloud, the infrastructure cloud, and the back-end cloud. Lee *et al.* combined the vehicular cloud computing and the information-centric network into a new architecture for VC networking [15]. Especially, Jang *et al.* proposed a software-defined vehicular cloud architecture to achieve flexible VC control and efficient resource utilization in a centralized manner [16].

Compared with VCC, task scheduling has been widely investigated in the MEC. Mao *et al.* investigated a green MEC system with energy harvesting devices, where a Lyapunov based dynamic computation offloading scheme was developed [17]. By jointly considering the offloading decision, the CPU-cycle frequency, and the transmit power allocation, the proposed scheme can achieve the tradeoff between the execution latency and task failure. Zhang *et al.* proposed a theoretical framework of energy-optimal mobile cloud computing under the stochastic wireless channel, and a threshold policy was derived for the execution strategy of applications with small output data [18]. Chen *et al.* investigated the computation offloading decision-making problem for the multi-user multi-channel environment, considering the tradeoff between the energy consumption and the execution delay [19]. The problem was further formulated as a multi-user computation offloading game, and a distributed computation offloading algorithm, which can achieve a Nash equilibrium, was developed. You *et al.* investigated the resource allocation problem in a multiuser MEC system based on time-division multiple access (TDMA) and orthogonal frequency-division multiple access (OFDMA) [20]. For the TDMA-based MEC system, the resource allocation is formulated as a convex optimization problem, while for the OFDMA system, it is formulated as a mixed-integer problem, which can be solved by a low-complexity sub-optimal algorithm. As for the vehicular cloud, Zheng *et al.* proposed an optimal computation resource allocation scheme to maximize the total long-term expected reward of the VCC system, of which the optimization problem can be formulated as an infinite horizon semi-Markov decision process (SMDP) [11]. Similarly, Lin *et al.* proposed an SMDP model for VCC resource allocation considering the heterogeneity of vehicles and roadside units [21].

According to the mobility of vehicles, the vehicular cloud can be divided into two categories, i.e., mobile and static, consisting of vehicles in movement and statically parked ones, respectively. For the static vehicular cloud, due to its relatively stable resources, it is more suitable for communication and safety services. On the other hand, the mobility of vehicles in mobile vehicular cloudlet (MVC) makes them proper for data ferrying

service [12]. However, it also brings some challenges to the VCC system, one of which is the instability of the onboard resources. To deal with this problem, some researches have been done on the simulation of vehicular mobility [22]–[25]. Akhtar *et al.* analyzed the VANET topology in a highway scenario by applying node degree, neighbor distribution, number of clusters, and link duration [22]. Ali *et al.* proposed a new mobility model for VANET in urban and suburban areas, concerning the topographic and socioeconomic characteristics of infrastructures and the spatiotemporal population distribution [23]. Wang *et al.* evaluated the performance of an MVC in practical environments through a real-world vehicular mobility trace of Beijing [24]. Zhang *et al.* leveraged the current vehicular velocity and the moving direction to estimate the contact duration of each vehicle in the cloud [25]. In addition to the instability, the inter-dependency of computing tasks is another challenge we need to deal with [26]–[28]. Zhang *et al.* considered a series of tasks executed sequentially, forming a linear topology [26]. Deng *et al.* proposed a genetic algorithm based offloading system to determine whether the services of a work-flow should be offloaded [27]. In [28], a directed acyclic task graph is applied to represent the relationship among tasks. Different from previous studies about the task scheduling in MEC and VCC, in this paper, we investigate the computation offloading problem in the vehicular cloud, with the challenges of instability, heterogeneity, and inter-dependency of computing tasks, which can provide the useful guideline to the efficient utilization of the onboard computing resources.

The remainder of the paper is organized as follows. Section II presents the system model and problem formulation. The MGA based scheduling scheme is proposed in Section III. The system stability analysis and rescheduling scheme are introduced in Section IV. Simulation results are given in Section V. Finally, Section VI concludes the paper and suggests our future works. The main notations used in this paper are listed in Table I.

II. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Overview

We consider a VC-assisted computation offloading scenario in a cellular network, where N vehicles $\mathcal{V} = \{v_1, \dots, v_N\}$ within the coverage of a base station (BS) are organized together to form a VC. In order to enhance the overall computing capacity, conventional edge cloud (EC) may offload part of its computing missions $\mathcal{M} = \{m_1, \dots, m_K\}$ to the VC within its coverage area. The service requests from EC are first sent to the BS, which is aware of the vehicular conditions, i.e., the arrival/departure time and the computing capability of each vehicle in VC. After receiving the requests, the BS will then assign these missions to VC for processing. Fig. 1 shows such a scenario, where 6 vehicles are organized together as a VC. According to the computing capabilities of these vehicles, a computation-intensive mission consisting of 4 dependent tasks is then assigned to some of these vehicles for processing. As shown in this figure, the first task is assigned to vehicle A. After finishing task 1, vehicle A is scheduled to transmit the output data of task 1 to vehicle B for the processing of task 2.

TABLE I
MAIN NOTATIONS

Notation	Definition
$\mathcal{V}$	Vehicle set: $\mathcal{V} = \{v_1, \dots, v_N\}$
$\mathcal{M}$	Missions offloaded to VC: $\mathcal{M} = \{m_1, \dots, m_K\}$
d_k	Number of tasks in mission $m_k \in \mathcal{M}$
λ_m	Arrival rate of computing missions
λ_v	Arrival rate of vehicles into VC
$[\delta_n, \mu_n]$	Contact interval of vehicle $v_n \in \mathcal{V}$
$\mathcal{B}^k$	Task set of mission $m_k \in \mathcal{M}$: $\mathcal{B}^k = \{b_1^k, \dots, b_{d_k}^k\}$
ϕ_i^k	Workload, input, output data of task b_i^k : $\phi_i^k = (\omega_i^k, \alpha_i^k, \beta_i^k)$
$\mathbf{I}^k$	Logical matrix of mission m_k : $\mathbf{I}^k = (I_{i,l}^k)_{d_k \times d_k}$
$p_{i,n}$	Processing time of task i executed on vehicle v_n
f_n	Computing capability of vehicle $v_n \in \mathcal{V}$
$c_{i,n,h}$	Communication time of task i processed on v_n in S_t^h
t_u	Unit time slot
S_t^h	Length of the h th scheduling time slot
γ	Scheduling parameter $\gamma \in N_+$
τ_h	End time of the h th scheduling slot S_t^h
t_i^R	Ready time of task i
t_i^G	Generation time of task i
$\varsigma_{i,n,h}$	Response time interval of task i processed in S_t^h
$\overline{D}$	Midrange task load of VC
p_o	Offline probability
ϵ	Task-load bound
$\mathcal{T}_n$	Incoming or outgoing task flow of vehicle $v_n \in \mathcal{V}$

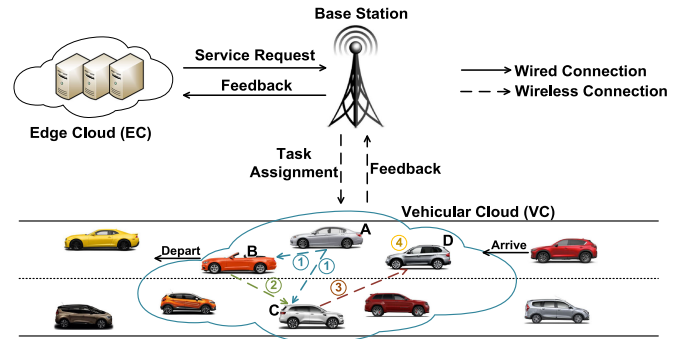


Fig. 1. Cooperative task scheduling for computation offloading in VC.

Specifically, due to the inter-dependency of these 4 tasks, task 3 needs the output data of both task 1 and 2. Hence, vehicles A and B will then transmit both output data to vehicle C for the processing of task 3. Then, task 4 is processed on vehicle D with the output of vehicle C, and it will finally feedback the result to the BS as the completion of the whole mission.

B. Vehicular Mobility Model

One of the challenges in VC is the instability of onboard resources caused by the mobility of vehicles, i.e., only in VC can vehicles be available for computation offloading. We assume the arrivals of vehicles into VC following a Poisson process with the arrival rate λ_v [11]. Since the vehicle is in the VC as long as it is within the coverage of the BS, we only focus on the cell dwell time (i.e., the cell residence time) of vehicles. Specifically, the Hyper-Erlang distribution [29] is utilized to model the cell dwell time of each vehicle. Due to the universal approximation capability, Hyper-Erlang models can be utilized to fit the field data, and then to characterize the cell dwell time of vehicles. The probability density function (PDF) of Hyper-Erlang distribution is given by

$$f_{x_d}(t) = \sum_{i=1}^n \frac{p_i \eta_i^l t^{l-1}}{(l-1)!} e^{-\eta_i t} \quad (1)$$

where $\sum_{i=1}^n p_i = 1$, n and l respectively represent the number of the stages and the number of phases in each stage, $\frac{1}{\eta_i}$ represents the mean permanence time (i.e., cell dwell time) of stage i , and p_i denotes the probability of choosing stage i , $i \in \{1, 2, \dots, n\}$. The expectation of Hyper-Erlang distribution can be given by

$$E[X_d] = \sum_{i=1}^n p_i l / \eta_i \quad (2)$$

We then define the contact interval of each vehicle in VC as the time interval, during which the vehicle is within the coverage of the BS, and it is defined as $[\delta_n, \mu_n]$, where δ_n and μ_n represent the arrival and departure time of v_n , respectively. To guarantee the availability of the computing resources, the offloaded computing missions need to be finished before the vehicles leave VC.

C. Task Model

We assume that each computation-intensive mission $m_k \in \mathcal{M}$ can be divided into several discrete offloadable tasks $\mathcal{B}^k = \{b_1^k, \dots, b_{d_k}^k\}$, where each task can be processed on any of the N vehicles in VC, and d_k represents the total number of tasks belonging to the mission m_k . More specifically, these tasks can be either independent of each other or mutually dependent. In the former case, all tasks can be offloaded simultaneously and processed in parallel. However, in the latter case, the mission is composed of tasks that need input from some others and simultaneous offloading may not be applicable. For instance, a video navigation application [30] can be divided into 4 parts: graphics, face detection, camera preview and video processing. The intra-dependency of the graphics is sequential, while that of the face detection is parallel. In this paper, we focus on the case with mutually dependent tasks, where the inter-dependency of tasks will affect the offloading decisions. Furthermore, we observe that the complex dependency can be divided into three basic logic topologies, i.e., linear, tree and mesh. As shown in Fig. 2, each of the three missions is divided into five tasks. To describe the parametric context of each task, we define a ternary tuple [26] as $\phi_i^k = (\omega_i^k, \alpha_i^k, \beta_i^k)$, where ω_i^k , α_i^k , and β_i^k

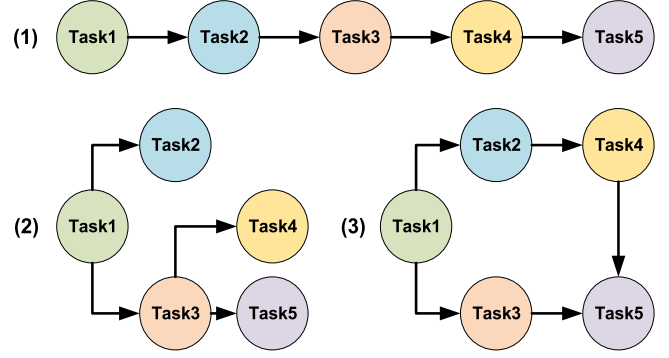


Fig. 2. Task-flow model.

represent the computation workload, the input data size, and the output data size of the i -th task of mission m_k , respectively. As such, the total computation workload of m_k is $\sum_{i=1}^{d_k} \omega_i^k$. As shown in Fig. 2(1), mission 1 is with linear logic, in which $\beta_i^k = \alpha_{i+1}^k$. In the mesh logic of mission 3 shown in Fig. 2(3), there exists $\beta_3^k + \beta_4^k = \alpha_5^k$, which means that task 5 can only be processed after the completion of tasks 3 and 4. Among these three logical topologies, linear logic topology is a single input structure with strong data dependency between tasks, while the data dependency of tree and mesh structures is looser compared to the linear logic model. Therefore, the inter-dependency of tasks should be considered when making offloading decisions. We use a lower triangular matrix $\mathbf{I}^k = (I_{i,l}^k)_{d_k \times d_k}$ to represent the inter-dependency of the tasks in mission $m_k \in \mathcal{M}$, where $I_{i,l}^k = \{0, 1\}$, $1 \leq l < i \leq d_k$. If task i needs the output data of task l , $I_{i,l}^k = 1$, otherwise $I_{i,l}^k = 0$. For instance, the logical matrix of mission 3 in Fig. 2 can be given by

$$\mathbf{I}^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \quad (3)$$

where $I_{5,3}^3, I_{5,4}^3 = 1$ means that task 5 needs the output data of tasks 3 and 4.

D. Execution Model

We assume that the arrival of computing missions follows a Poisson process with the arrival rate λ_m [11]. To calculate the response time of a certain mission, we need to first model the execution process of every single task. Specifically, the response time of each task consists of three parts, i.e., processing time, queuing time, and communication time.

- **Processing Time:** We consider that vehicles may have different computing capabilities such as different clock frequencies. The processing time of task i processed on vehicle n is then given by

$$p_{i,n} = \omega_i f_n^{-1} \quad (4)$$

where ω_i represents the computation workload of task i , and f_n denotes the processing capability (i.e., clock frequency) of vehicle v_n .

- **Queuing Time:** When a task is assigned to a certain vehicle, it may not be processed immediately, due to the existing task being processed in that vehicle. The amount of time it takes for a task to wait for being processed is defined as the queuing time. To explain the queuing time, we first introduce the following definitions.

Definition 1. Unit Time Slot: To fit the characteristic of the scenario, we define the unit time slot of the vehicular cloud computing system as the minimum processing time, which is given by

$$t_u = \omega_{\min} f_{\max}^{-1} \quad (5)$$

where $\omega_{\min}$ is the minimum computation workload of all the offloaded tasks, and $f_{\max}$ represents the maximum clock frequency of the vehicles in VC. The unit time slot t_u is used as a normalize time for the computing task scheduling problem.

Definition 2. Scheduling Time Slot: The scheduling slot is defined as the time duration consisting of the execution and data transmission process. Considering the variety of the computation workload and capabilities, we divide the time horizon into discrete scheduling time slots, where the time duration of the h -th scheduling time slot ($h \in \{1, 2, \dots, H\}$) is given by

$$S_t^h = \begin{cases} \gamma t_u, & h \equiv 0 \pmod{\gamma} \\ \text{rem}(h, \gamma) t_u, & \text{otherwise} \end{cases} \quad (6)$$

where H is the maximum number of scheduling time slots, and $\gamma \in N_+$ is defined as the scheduling parameter of VC, which is designed according to the vehicular computing capabilities and computation workload of tasks, and $\text{rem}(h, \gamma)$ is the remainder of h divided by γ . We can then get the end time of the h -th scheduling time slot S_t^h given by

$$\tau_h = t_u \left(\lfloor h/\gamma \rfloor \sum_{l=1}^{\gamma} l + \sum_{k=0}^{\text{rem}(h, \gamma)} k \right) \quad (7)$$

where $\lfloor h/\gamma \rfloor$ denotes the integer part of h divided by γ . The maximum possible time duration to complete all tasks can be given by

$$T_{\max} = \sum_{h=1}^H S_t^h \quad (8)$$

We introduce the aforementioned definition for the time duration of each scheduling time slot in order to reduce the complexity of the task scheduling problem.

Definition 3. Ready Time: Considering the existence of multiple inputs, the ready time of a task i is defined as the earliest time when all the preliminary tasks of this task are completed [28], which can be given by

$$t_i^R = \max \{ \tau_h I_{i,l} x_{l,n,h} \mid 1 \leq l < i, h \leq H \} \quad (9)$$

where $x_{l,n,h} = \{0, 1\}$ indicates whether task l is processed on vehicle v_n in the h th scheduling slot.

Definition 4. Queuing Time: The queuing time for task i processed on vehicle v_n in S_t^h is defined as the time duration from the ready time of the task to the time instant that the task starts to be processed, which is given by

$$q_{i,n,h} = \begin{cases} \tau_{h-1} - t_i^G & i = 1 \\ \tau_{h-1} - t_i^R, & i > 1 \end{cases} \quad (10)$$

where t_i^G represents the generation time of the first task (i.e., the whole computation mission). Note that the queuing time of a task is an integral multiple of the unit time slot t_u since the scheduling scheme is based on t_u .

- **Communication Time:** In case of multiple inputs, when calculating the input/output data transmission time among different vehicles, we need to consider the logical matrix of the related missions. The communication time of task i processed on vehicle n during the h -th scheduling time slot is defined as the maximum communication time of the previously required tasks, which can be given by

$$c_{i,n,h} = \max \left\{ \frac{\beta_l}{r_h} I_{i,l} \mid 1 \leq l < i \right\} \quad (11)$$

where r_h represents the data transmission rate between the corresponding vehicles during the h th scheduling time slot, and I_i is the dependency matrix of task i .

E. Problem Formulation

With the analysis in Section II-D, we define the response interval of task i as the time interval between the ready time and the end time of the scheduling slot S_t^h , which can be given by

$$\varsigma_{i,n,h} = (\tau_{h-1} - q_{i,n,h}, \tau_h] \quad (12)$$

Then, the response time of the k -th mission is the length of the union of these time intervals

$$T_{res}^k = \left| \bigcup_{i=1}^{d_k} \varsigma_{i,n,h} \right| \quad m_k \in \mathcal{M}, v_n \in \mathcal{V}, h \leq H \quad (13)$$

To simplify the problem, we can rewrite (13) as

$$T_{res}^k = \max_{b_i^k \in \mathcal{B}^k, v_n \in \mathcal{V}, h \leq H} \{ \tau_h x_{i,n,h} - t_k^G \} \quad (14)$$

where the response time of each mission is defined as the maximum duration between the generation time and the scheduling time of the tasks. Note that t_k^G is a constant, and thus we simplify the objective to minimize the average mission completion time. The scheduling problem can be given by

$$\min_{\{x_{i,n,h}\}} \quad \frac{1}{K} \sum_{k=1}^K \max_{b_i^k \in \mathcal{B}^k, v_n \in \mathcal{V}, h \leq H} \{ \tau_h x_{i,n,h} \} \quad (15)$$

$$\text{s.t.} \quad \sum_{n=1}^N \sum_{h=1}^H x_{i,n,h} = 1 \quad \forall b_i^k \in \mathcal{B}^k, \forall m_k \in \mathcal{M} \quad (16)$$

$$x_{i,n,h} = 0 \quad \text{if} \quad S_t^h < c_{i,n,h} + p_{i,n,h} \quad (17)$$

$$x_{i,n,h} = 0 \quad \text{if} \quad (\tau_{h-1}, \tau_h] \not\subset [\delta_n, \mu_n] \quad (18)$$

$$q_{i,n,h} \geq 0 \quad (19)$$

$$x_{i,n,h} = \{0, 1\} \quad (20)$$

where $x_{i,n,h}$ indicates whether task $b_i^k \in \mathcal{B}^k$ is scheduled to process on vehicle v_n within the h -th scheduling slot S_t^h . We use the interval $(\tau_{h-1}, \tau_h]$ as the approximation of the actual communication and processing time of task b_i^k . The objective (15) is to minimize the average completion time. Constraint (16) ensures that each task is scheduled only once. Constraint (17) ensures that each task i completed by time τ_h on vehicle v_n must satisfy that the scheduling slot is larger than the sum of the communication and processing time. Constraint (18) guarantees the availability of the computing resources. Constraint (19) refers to the inter-dependency of tasks in each mission. Since the problem has an integer constraint (Constraint (20)), according to [31], the task allocation problem can be reduced to a scheduling problem, which is NP-hard. Therefore, there is no polynomial time algorithm to find the optimal solution. Detailed solution to this problem will be introduced in the next section.

III. MGA BASED JOINT SCHEDULING SCHEME

In this section, an MGA based scheduling scheme is proposed to solve the NP-hard task scheduling problem. Specifically, we modify 3 parts (i.e., Encoding, Crossover, and Mutation) of the traditional genetic algorithm to reduce the complexity of the problem.

A. Algorithm Overview

A genetic algorithm usually begins with a population of candidate solutions, which can be evolved toward better solutions to a certain optimization problem. Generally, each of these candidate solutions is a combination of properties (called chromosomes or genotype), which can be represented in binary. The genetic algorithm is an iterative process. During the first iteration, a population of feasible solutions is randomly generated. In each iteration, the fitness (i.e., the objective of the optimization problem) of each individual is evaluated. Based on the fitness, the more fit individuals are stochastically selected from the current population while some individuals' chromosomes are recombined or mutated to form a new set of individuals. The newly generated solutions are then used in the next iteration of the algorithm. Commonly, the iterative process terminates when a maximum number of iterations or a satisfactory fitness level has been reached.

As for the NP-hard task scheduling problem, the application of ordinary genetic algorithm will cause high computational complexity. Therefore, we modified some parts of the algorithm to reduce the complexity of the problem. Detailed modifications are given as follows.

B. Encoding

In traditional genetic algorithms, a standard representation of each candidate solution is an array of bits (i.e., 0 or 1). Binary coding has the advantages of high stability and large population

Algorithm 1: MODIFIED GENETIC ALGORITHM BASED JOINT SCHEDULING SCHEME.

Input: $\{\{\delta_n, \mu_n\}, f_n\}_{v_n \in \mathcal{V}}$
 $\{\Phi^k = \{\phi_i^k \mid i = 1, \dots, d_k\}, \mathbf{I}^k\}_{m_k \in \mathcal{M}}$
 $Round_{stop}$, $population$,
 $P_{Crossover}$, $P_{Mutation}$

Output: $\mathbf{A}^* = \left\{ \theta_i^* \mid i = 1, \dots, \sum_{l=1}^K d_l \right\}$

```

1 Take  $t = 0$ 
2 Initialization:
    $\mathbf{A}^t = \left\{ \theta_{i,j}^t \mid i = 1, \dots, \sum_{l=1}^K d_l, j = 1, \dots, population \right\}$ 
3 while  $t \leq Round_{stop}$  do
4    $\mathbf{F}^t = \{\text{Fitness}(\mathbf{A}_j^t) \mid j = 1, \dots, population\}$ 
5   Descending Sort:  $\mathbf{F}^t, \mathbf{A}^t$ 
6    $\mathbf{A}_{elite}^t = \min_{\mathbf{A}_j^t \in \mathbf{A}^t} \text{ResponseTime}(\mathbf{A}_j^t)$ 
7   for  $j = 1 : population$  do
8     Roulette Selection:  $\mathbf{A}_F^t, \mathbf{A}_M^t \in \mathbf{A}^t$ 
9     Take  $poset = []$ 
10    for  $poscut = 1 : K - 1$  do
11       $pos = \sum_{l=1}^{poscut} d_l$ 
12       $\mathbf{A}_O^t = \text{Crossover}(\mathbf{A}_F^t, \mathbf{A}_M^t, pos)$ 
13      if  $\text{Feasibility}(\mathbf{A}_O^t) = 1$  then
14         $poset = poset \cup pos$ 
15     $p_1 = \text{rand}()$ 
16    if  $p_1 \leq P_{Crossover} \cap poset \neq []$  then
17       $\forall pos \in poset$ 
18       $\mathbf{A}_j^{t+1} = \text{Crossover}(\mathbf{A}_F^t, \mathbf{A}_M^t, pos)$ 
19       $p_2 = \text{rand}()$ 
20      if  $p_2 \leq P_{Mutation}$  then
21         $\mathbf{A}_j^{t+1} = \text{Mutation}(\mathbf{A}_j^{t+1})$ 
22        if  $\text{Feasibility}(\mathbf{A}_j^{t+1}) = 0$  then
23           $\mathbf{A}_j^{t+1} = \text{Modification}(\mathbf{A}_j^{t+1})$ 
24      else
25         $\mathbf{A}_j^{t+1} = \mathbf{A}_F^t$ 
26     $t = t + 1$ 
27  $\mathbf{A}^* \leftarrow \mathbf{A}_{elite}^t$ 

```

diversity. However, in this problem, the offloading decision is a combination of scheduling slots and vehicles, if binary coding is applied, the dimension of each individual candidate solution will be $NH \sum_{k=1}^K d_k$, where N and H respectively represent the number of vehicles and scheduling slots, and d_k is the number of tasks in mission m_k . The required large storage space will reduce the efficiency of the algorithm, and the decoding process can be difficult to implement. To this end, the integer encoding is employed, in which the chromosome of each candidate solution can be represented by an integer $\theta \in \{1, 2, 3, \dots, NH\}$. Then, the dimension of each solution is reduced to $\sum_{k=1}^K d_k$, which can improve the efficiency of the algorithm. The decoding process is simple. Given an integer θ , the index of the scheduled vehicle

is $\lceil \theta/H \rceil + 1$, and the assigned scheduling slot can be given by

$$S_t^h = \begin{cases} S_t^H, & \text{rem}(\theta, H) = 0 \\ S_t^{\text{rem}(\theta, H)}, & \text{otherwise} \end{cases} \quad (21)$$

where $\text{rem}(\theta, H)$ is the remainder of θ divided by H .

C. Crossover

For each iteration, individuals with high fitness will be stochastically selected to perform the operation of crossover. The crossover operation is to combine two candidate solutions (i.e., parents) with the possibility of generating higher fitness offspring. Usually, a random single point in parents' chromosomes is chosen to generate new offspring chromosomes through exchanging genes around this point. However, in the task scheduling problem, the random crossover operation may generate new offsprings, which may not be feasible to the original constraints. The fix operation for infeasible offsprings will then increase the complexity of the algorithm. To guarantee the feasibility of the newly generated solutions, we give the definition of relatives.

Definition 5. Relatives: For two selected individuals A_1 and A_2 , randomly choose $p \in \{1, 2, 3, \dots, K\}$. We can then get the exchange point as $p^c = \sum_{l=1}^p d_l$. The offspring of A_1 and A_2 based on p^c is represented by $A_{1,2}^{p^c}$. If $A_{1,2}^{p^c}$ is infeasible, A_1 and A_2 are relatives based on p^c . Therefore, if the selected individuals are relatives for a certain exchange point, the crossover operation cannot be performed. As a result, we can guarantee the feasibility of the offsprings.

D. Mutation

To improve the fitness of the candidate solutions as well as avoid early convergence, a slight modification of chromosomes is needed. This operation is called mutation, of which a gene of an individual is randomly changed. However, this will also lead to an infeasible solution. A modification function is then designed to guarantee the feasibility of the mutated individuals.

E. MGA Based Scheduling Scheme

The designed algorithm is given in Algorithm 1. For each iteration, the individuals will be resorted according to their fitness (lines 4-5). Individuals with the highest fitness will be retained until the next generation (line 6). For the stochastically selected individuals, if they are not relatives, there is a possibility of generating new individuals to form the population of the next iteration (lines 8-18). The iterations will proceed until the optimal solution is achieved, or until some convergence criteria is achieved. Additionally, considering the dichotomy and the relatives detection (lines 10-14) applied in this algorithm, the computational complexity of each iteration is $\mathcal{O}(K \log N + KN)$, where N and K denote the population size and number of the offloaded missions, respectively. Therefore, the computational complexity of this algorithm is $\mathcal{O}(GNK)$, where G represents the number of iterations.

IV. STABILITY ANALYSIS AND RESCHEDULING SCHEME

The VCC system is a distributed computing system, where the uneven distribution of the computation workload may cause some vehicles overloaded, which will further reduce the system stability. In this section, we analyze the stability of the VCC system. Specifically, we propose a statistical priority based rescheduling algorithm to enhance the offloading decision made by the MGA based scheme. The basic idea of this rescheduling scheme is to find the midrange task load $\bar{D}$ first, and then determine the set of tasks of each overloaded vehicle, which should be redirected to other vehicles such that the average task load of VC is approximately $\bar{D}$, which can increase the system stability.

A. Stability Analysis

One of the challenges of VC is the instability caused by the mobility of vehicles. Furthermore, even though some vehicles are within VC, they can be temporarily or permanently unavailable for computation offloading, and this will further reduce the system stability. The system stability is defined as the mission response rate, i.e., the ratio of the number of completed missions to the number of offloaded missions. We define that a vehicle within VC is offline if it is unavailable for computation offloading. We consider extreme conditions, where vehicles may be permanently offline in case of overload. The tasks assigned to the offline vehicles will be lost, resulting in the reduction of the mission response rate. The offline probability of each vehicle v_n in each unit time slot is defined as $p_o(D_n)$, where D_n represents the task load of vehicle v_n . We can then get the offline probability of vehicle v_n in the j th slot as $(1 - p_o(D_n))^{j-1} p_o(D_n)$, which follows the geometric distribution. Particularly, we assume that the offline probability p_o and the task load D_n is positively related. Suppose that the task assignment for vehicle v_n is

$$\mathcal{A}_n^* = \left\{ (S_{t,n}^h, b_l^{k,n}) \mid \delta_n \leq \tau_{h-1}, \tau_h \leq \mu_n, 1 \leq l \leq d_k \right\} \quad (22)$$

where $S_{t,n}^h$ and $b_l^{k,n}$ are defined as the assigned scheduling slot and the task executed in that slot. If v_n is offline in the j th slot, the task set $\mathcal{A}_{n,j}^* = \left\{ (S_{t,n}^h, b_l^{k,n}) \mid j t_u \leq \tau_{h-1} \leq \mu_n \right\}$ will be lost. This will lead to the failure of the missions consisting of these lost tasks. The expectation of the total number of failed missions of v_n can be given by

$$M_n = \left\langle \sum_{j=1}^{\mathcal{M}} \{1 - p_o(D_n)\}^{j-1} p_o(D_n) |\mathcal{A}_{n,j}^*| \right\rangle \quad (23)$$

where $\langle \cdot \rangle^{\mathcal{M}}$ is to calculate the exact number of the failed missions in case of the duplication of the tasks belonging to the same mission $m_k \in \mathcal{M}$. The response rate can then be given by

$$R_{res} = 1 - K^{-1} \left\langle \sum_{n=1}^N M_n \right\rangle^{\mathcal{M}} \quad (24)$$

Through (24), we can see that for a certain vehicle v_n , the larger the amount of $\mathcal{A}_{n,j}^*$ (i.e., task load) is, the more missions may be failed. Therefore, evenly distributing tasks can effectively

increase the mission response rate, thereby enhancing the stability of the VCC system.

B. Task Load of Vehicular Cloud

Since the objective is to minimize the average response time of the computing missions as in (15), most of the tasks will be assigned to the vehicles with strong computing capabilities. In this case, the proposed MGA based scheme cannot guarantee the even distribution of tasks, which may reduce the system stability. Therefore, it is necessary to redirect some tasks such that each vehicle has the similar task load. We design an approach to identify the midrange task load among the vehicles and decide the out-going tasks of each overloaded vehicle and in-coming tasks of each underloaded vehicle.

The task load of vehicle v_n is defined as the ratio of the occupation time to the contact duration, which can be given by

$$D_n = \left[\sum_{k=1, b_i^k \in \mathcal{B}^k}^K \sum_{h=1}^H (c_{i,n,h} + p_{i,n,h}) x_{i,n,h} \right] (\mu_n - \delta_n)^{-1} \quad (25)$$

Define the midrange task load $\bar{D}$ as $(D_{\min} + D_{\max})/2$, where $D_{\max} = \max \{D_n | v_n \in \mathcal{V}\}$ and $D_{\min} = \min \{D_n | v_n \in \mathcal{V}\}$. All the vehicles can then be partitioned into two disjoint sets, the set $\mathcal{V}_O$ of overloaded vehicles:

$$\mathcal{V}_O = \{v_j | D_j > \bar{D}\} \quad (26)$$

and the set $\mathcal{V}_U$ of underloaded vehicles:

$$\mathcal{V}_U = \{v_m | D_m \leq \bar{D}\} \quad (27)$$

For each overloaded vehicle $v_j \in \mathcal{V}_O$, we need to determine the outgoing task flow $\mathcal{T}_j^-$ that should be redirected from v_j such that the task load of v_j is within the task load bound ϵ of $\bar{D}$, i.e., we need to find a $\mathcal{T}_j^-$ such that

$$|\bar{D} - D_j(\mathcal{T}_j^-)| \leq \epsilon \quad (28)$$

Similarly, for each underloaded vehicle $v_m \in \mathcal{V}_U$, we need to determine the incoming task flow $\mathcal{T}_m^+$ that should be redirected to v_m such that the task load of v_m is within an acceptable bound of $\bar{D}$, i.e.,

$$|\bar{D} - D_m(\mathcal{T}_m^+)| \leq \epsilon \quad (29)$$

C. Statistical Priority Based Rescheduling Scheme

The objective of the rescheduling scheme is to find the proper outgoing task sets $\mathcal{T}^-$ and incoming task sets $\mathcal{T}^+$ of each vehicle to enhance the stability of the VCC system while minimizing the average response time. Note that the offloading decision made by the MGA based scheme consists of 2 parts, i.e., the scheduling time slot of each task and the processing vehicles for each scheduling slot. Therefore, to guarantee the average response time, we only need to reschedule the vehicles of each scheduling slot. For a certain scheduling slot, each vehicle in VC will have 2 states (i.e., occupied or unoccupied). For each occupied vehicle $v_j \in \mathcal{V}_O$, a decision needs to be made, i.e., whether the task needs to be redirected. Moreover, if the task is decided to be

Algorithm 2: SP BASED RESCHEDULING SCHEME.

Input: Offloading decision $\mathcal{A}^*$ from MGA based scheme, task-load bound $\epsilon, \{\delta_n, \mu_n\}, f_n\}_{v_n \in \mathcal{V}}$, $Round_{stop}$

Output: Task flow $\mathcal{T}_n$ for each vehicle $v_n \in \mathcal{V}$

```

1 Calculation:  $D_n, \forall v_n \in \mathcal{V}, D_{\max}, D_{\min}$ 
2 Set:  $\bar{D} = (D_{\min} + D_{\max})/2, \mathcal{T}_n = [], \forall v_n \in \mathcal{V}$ 
3 Classification:  $\mathcal{V}_O = \{v_j | D_j > \bar{D}\}$ 
    $\mathcal{V}_U = \{v_m | D_m \leq \bar{D}\}$ 
4 Set:  $Pr_j = D_j - \bar{D}$ 
    $Th_m = D_m$ 
5 Take  $\lambda = 0, t = 1$ 
6 while  $\lambda = 0 \cap t \leq Round_{stop}$  do
7   for each  $S_t^h \in \mathcal{A}^*$  do
8     Find: Task Set:  $\mathcal{T}_O^h$ , Vehicle Set:  $\mathcal{V}^h \subseteq \mathcal{V}_U$ 
9     for each task  $j \in \mathcal{T}_O^h$  do
10      RandomSelect:  $v_m \in \mathcal{V}^h$ 
11      if  $Pr_j \geq Th_m \cap f_m \geq$ 
12         $f_j \cap (v_m \text{ is unoccupied})$  then
13         $\mathcal{T}_j^- = \mathcal{T}_j^- \cup \text{task } j$ 
14         $\mathcal{T}_m^+ = \mathcal{T}_m^+ \cup \text{task } j$ 
15        Update:  $D_j, D_m, Pr_j, Th_m$ 
16        Status of  $v_m$  in  $S_t^h$ : Occupied.
17   for each  $v_n \in \mathcal{V}$  do
18     if  $D_n \in [\bar{D} - \epsilon, \bar{D} + \epsilon]$  then
19        $\lambda_n = 1$ 
20     else
21        $\lambda_n = 0$ 
22    $\lambda = \bigcap_{n=1}^N \lambda_n$ 
    $t = t + 1$ 
```

redirected, we then need to deal with the issue of allocating an unoccupied vehicle $v_m \in \mathcal{V}_U$ to process this redirected task. We propose a statistical priority (SP) based rescheduling scheme to determine the task flow of each vehicle. The detailed scheme is given in Algorithm 2. For each scheduling slot S_t^h , each task processed on vehicle $v_j \in \mathcal{V}_O$ is assigned a different priority Pr_j . The priority is determined by the task load D_j , which is set to $Pr_j = D_j - \bar{D}$. Similarly, each vehicle $v_m \in \mathcal{V}_U$ is also assigned a threshold Th_m , which is set to $Th_m = D_m$. The tasks from overloaded vehicles can then be randomly redirected to the unoccupied underloaded vehicles of the same scheduling slot. The consequence of a certain task redirection is determined by the priority Pr_j , the threshold Th_m , and the vehicular computing capabilities. If Pr_j is higher than Th_m , and f_m is greater or equal to f_j , the task from v_j can be redirected to v_m , otherwise, the task cannot be redirected to the target vehicle. After each task redirection, the task load of each vehicle is updated, as well as the priority and threshold. In this case, for the overloaded vehicles $\mathcal{V}_O$, the higher the task load is, the more likely for the task outgoing, while for the underloaded vehicles $\mathcal{V}_U$, the lower the task load is, the more likely for the task

TABLE II
SIMULATION PARAMETERS

Parameters	Value
Computing capabilities of each vehicle	$f_i = f \text{ or } 2f$
Computation workload of each task	$\omega_i = \omega, 2\omega \text{ or } 3\omega$
Arrival rate of vehicles into VC	$\lambda_v = 0.5 \sim 2.5$
Unit time slot	$t_u = \omega(2f)^{-1}$
Scheduling parameter	$\gamma = 4$
Offline possibility of each vehicle	$p_o(D_n) = 0.02D_n$
Task load bound	$\epsilon = 0.1 \sim 0.4$

incoming. Consequently, the task load of each vehicle can be guaranteed to be bounded by $[\bar{D} - \epsilon, \bar{D} + \epsilon]$.

V. SIMULATION RESULTS

In this section, we conduct extensive simulations to evaluate the performance of the proposed MGA based scheduling scheme for computation offloading in VC. We compare the proposed scheme with the Greedy based scheme, which focuses on the local minimization of the response time of every single computing mission. To simplify the complex execution process, we assume that the communication time between vehicles is within a unit time slot t_u . The main parameters used in our analysis are provided in Table II. The scheduling parameter γ applied in the simulation is set to 4, which means that the length of each scheduling time slot $S_t^i, i = \{1, 2, 3, \dots, H\}$ is from $1t_u$ to $4t_u$. Considering the heterogeneity of vehicular computing capabilities, tasks with the workload of 3ω cannot be assigned to the vehicles with the computing capability of f , due to the fact that the largest scheduling slot is $4t_u = 2\omega f^{-1}$, which is smaller than $3\omega f^{-1}$.

A. Vehicle and Mission Simulation

In order to illustrate the computation offloading process in the vehicular cloud, we randomly select a scenario from the simulation, where 15 vehicles are organized together as a VC to offload the computing missions of the nearby EC. Fig. 3 shows the movement of these vehicles within a duration of $36 t_u$ (i.e., 15 consecutive scheduling time slots S_t).

It is observed that the contact interval of each vehicle within the coverage of the BS is different, which reflects the instability of these onboard computing resources in VC. In order to guarantee the completion of the offloaded missions, the tasks should be finished within the contact intervals of the processing vehicles.

Fig. 4 shows the logical topologies, arrival time, and computation workload of the offloaded computing missions. It is observed that 5 missions are offloaded to VC during the simulation period. The first mission is offloaded at $2t_u$, and there are 2 missions offloaded to VC at $3t_u$ and $4t_u$, respectively. Each mission is divided into 4 tasks with different computation

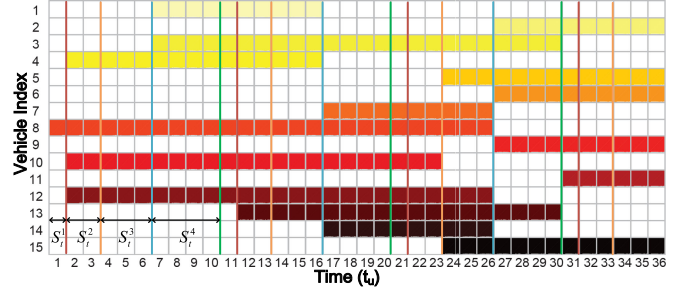


Fig. 3. Contact duration of each vehicle in VC.

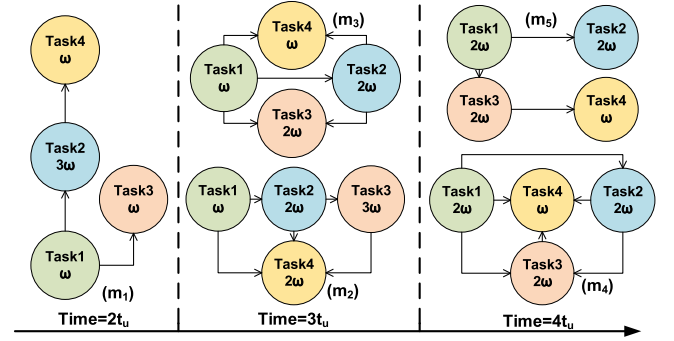


Fig. 4. Computation missions offloaded to VC.

workload. The structures of these missions are composed of the three basic logic topologies mentioned in Section II-C. Mission m_1 and m_5 are single input structures, while the other three missions are multiple inputs structures. Both the instability of resources (Fig. 3) and inter-dependency of tasks (Fig. 4) will affect the offloading decisions. In the following subsections, we will show the performance of the proposed MGA based scheme in terms of offloading decision, response time, and system stability.

B. Offloading Decisions

Fig. 5 shows the offloading decisions made by the Greedy and MGA based schemes. It can be seen from Fig. 5(a), by the Greedy scheme, the VC system always allocates the earliest scheduling slots to achieve the shortest response time of each mission. However, the Greedy scheme lacks the global concept, which may result in a local optimum. Compared with the MGA scheme in Fig. 5(b), the response time of missions m_1 and m_3 acquired from both schemes are the same, while the response time of m_2 of MGA scheme is larger than that of the Greedy scheme. As for missions m_4 and m_5 , the result of MGA scheme is superior to that of the Greedy scheme. From Fig. 5(a), we can see that even though the response time of m_2 of the Greedy scheme is shorter than that of the MGA scheme, the resource assignment for m_2 makes the resource unavailable for m_4 and m_5 , which leads to a longer queuing time of these two missions.

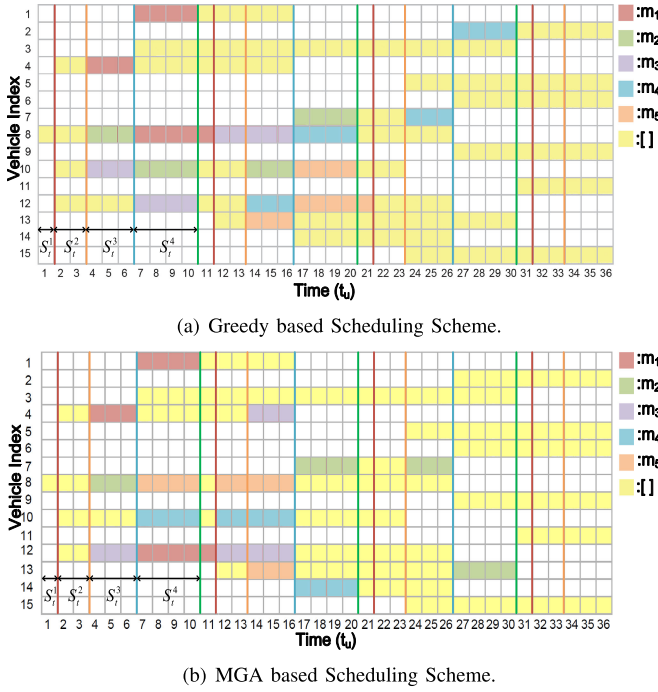


Fig. 5. Offloading decisions acquired from Greedy and MGA Scheduling Schemes.

Fig. 6(a) shows the efficiency of the proposed MGA based scheduling scheme. The average response time acquired by the Greedy scheme is $16t_u$, while the result of the MGA based scheme is $14t_u$ reduced by 12.5%. To further demonstrate the efficiency of the proposed MGA based scheme, Fig. 6(b) shows the queuing, communication, and processing time of the tasks in m_4 , which is the most complicated one compared with the others (Fig. 4). For the MGA based scheme, the first task b_1^4 is assigned to vehicle v_{10} in the 4th scheduling slot S_t^4 . Therefore, the queuing time $q_{1,10,4}$ is $\tau_3 - t_1^G + t_u = 3t_u$. The first task is transmitted from the BS to v_{10} , and thus the communication time $c_{1,10,4}$ of b_1^4 is t_u . Considering the computing capability and workload of v_{10} and b_1^4 , the processing time $p_{1,10}$ is $2\omega/(2f) = 2t_u$. After the completion of b_1^4 , the second task b_2^4 is processed on v_{10} in S_t^6 . The computation workload of b_2^4 is 2ω , and thus the processing time is also $2t_u$, which is larger than the length of $S_t^5 = t_u$. Therefore, the queuing time of the second task is $2t_u$. These 2 tasks are processed in the same vehicle v_{10} , and thus the communication time is 0. The third task of m_4 is also assigned to v_{10} in S_t^7 . Therefore, the queuing and communication time equals to 0, while the processing time is $2t_u$. Finally, the last task b_4^4 is scheduled to v_8 in S_t^8 . The queuing, communication, and processing time are the same t_u . As for the Greedy based scheme, due to resource assignment of the former 3 missions, the first task of m_4 is scheduled to v_7 in S_t^{12} , which leads to the queuing time of $10t_u$. The second task is assigned to a different vehicle v_8 , and thus there exists the communication time of t_u . Notably, the last task is assigned to v_2 , whose computing capability is f . Therefore, the processing time of the 4th task is $2t_u$. From the execution process of m_4 , we can see that the

proposed MGA scheduling scheme is superior to the Greedy scheme in terms of average response time.

C. Average Response Time

Additionally, we consider a scenario, where the mission status and the dwell time of each vehicle in VC are fixed. Fig. 7(a) shows that with the increase of the maximal number of vehicles that the VC can support, the average response time of the offloaded missions is decreased for both schemes, because the total resources in the VC are increasing. Meanwhile, a similar trend can also be found when the arrival rate of vehicles into VC is increased as shown in Fig. 7(b). When the number of scheduling time slots H is fixed, with the increase of the vehicle arrival rate, the number of vehicles per time slot in the VC is increased, as well as the computing resources. Fig. 7(c) shows that when the vehicular conditions, i.e., the arrival/departure time and the computing capability of each vehicle in VC are fixed, the average response time is increased with the number of the computing tasks offloaded. Furthermore, Fig. 8 shows that with the increase of the vehicle arrival rate, the processing efficiency of the offloaded missions increases for both schemes, where the processing efficiency is defined as the ratio of communication and processing time to total response time, which can be given by

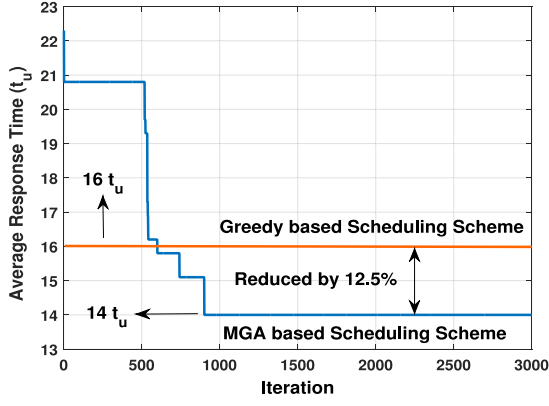
$$\xi = \left[\sum_{k=1}^K \sum_{b_i^k \in B^k} (c_{i,n,h} + p_{i,n,h}) \right] / \sum_{k=1}^K T_{res}^k \quad (30)$$

It can be seen that with the increase of λ_v , the queuing time is decreased, while more time is spent in the service queue of the Greedy based scheme compared with the MGA based scheme due to the lack of global optimality.

D. System Stability

Due to the heterogeneity of vehicular computing capabilities, tasks can be more likely assigned to vehicles with strong computing capabilities, which will reduce the stability of the VCC system. We design an SP-based rescheduling scheme to achieve the even distribution of computing tasks.

Given the offloading decision $\mathcal{A}^*$ from MGA scheme, we can acquire the task load of each vehicle from (25). Fig. 9 shows the task load of each vehicle before and after the rescheduling scheme. Note that vehicles v_2 and v_{4-6} are not shown in this figure because they are not involved in VC for the weak computing capabilities and short dwell time. We can see that v_7 has the maximum task load of $D_7 = 0.8$, while v_8 and v_9 have the similar task load of $D_8 = 0.7857$ and $D_9 = 0.7368$, respectively. Hence, these three vehicles have a higher chance of offline. According to the SP based rescheduling scheme, we find the minimum task load $D_{14} = D_{15} = 0$. The midrange task load $\bar{D}$ is then set to $(D_{\max} + D_{\min})/2 = 0.4$. The task load bound ϵ is set to 0.2. With the rescheduling scheme, we get the rescheduled resource assignment $\mathbf{A}_R^*$. It is observed that through rescheduling, the task load of v_7 , v_8 and v_9 is reduced from 0.8 to 0.3333, from 0.7857 to 0.5714, and from 0.7368



(a) Average response time of MGA and Greedy schemes.

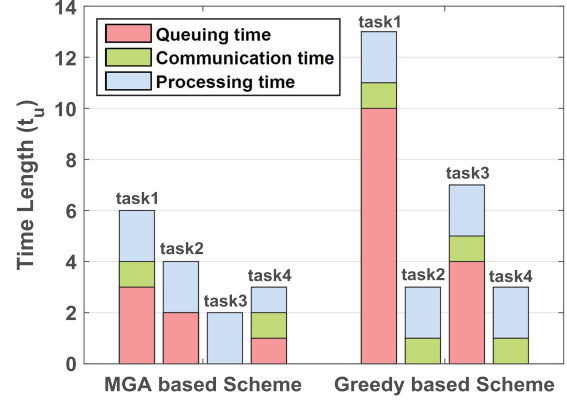
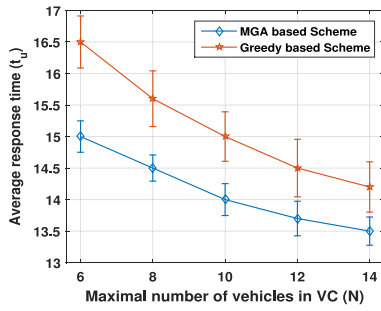
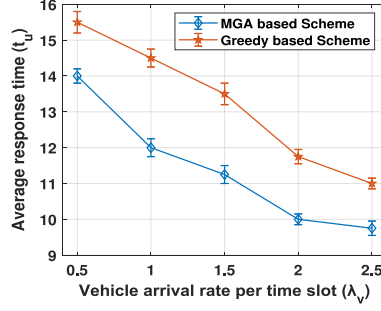
(b) Queuing, Communication, and Processing time of m_4 .

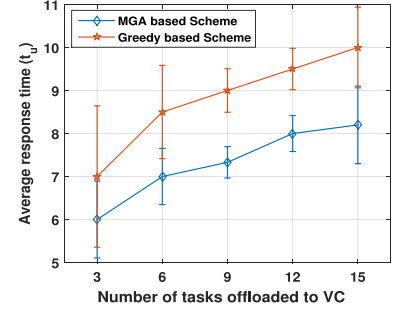
Fig. 6. Comparison of the MGA and Greedy based scheduling scheme.



(a) Average response time of different number of vehicles.



(b) Average response time of different vehicle arrival rate.



(c) Average response time of different number of offloaded tasks.

Fig. 7. Comparison of average response time of the MGA and Greedy based scheduling scheme.

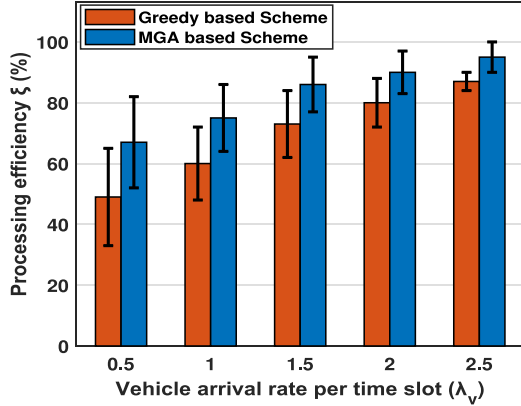
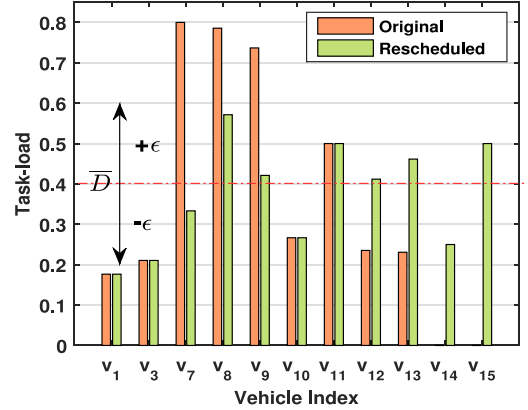


Fig. 8. Processing efficiency of different vehicle arrival rate.

Fig. 9. Task-load before and after rescheduling ($\epsilon = 0.2$).

to 0.4211, respectively. Detailed task rescheduling process is shown in Table III.

Fig. 10 shows the relationship between the mission response rate and the task load bound ϵ . It can be seen that the proposed SP based rescheduling scheme can help to increase the system stability. With the increase of ϵ , the allowable range of task load becomes larger, which leads to the reduction of the mission response rate. When $\epsilon = 0.4$, only the tasks of v_7 is rescheduled resulting in the response rate of 0.7482, which is still higher than that of the original MGA scheme.

TABLE III
TASK RESCHEDULING

Task Index	Scheduling Slot	Task Shifting Direction
b_2^1	S_t^7	$v_8 \rightarrow v_{13}$
b_3^1	S_t^8	$v_7 \rightarrow v_{15}$
b_3^2	S_t^7	$v_7 \rightarrow v_{12}$
b_1^4	S_t^7	$v_9 \rightarrow v_{14}$
b_1^5	S_t^{11}	$v_9 \rightarrow v_{15}$

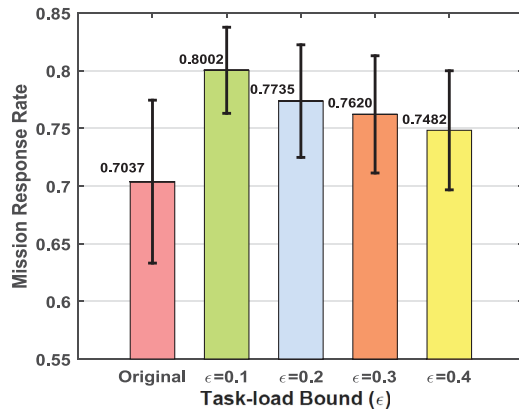


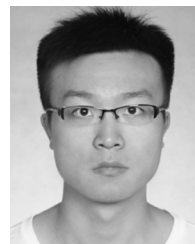
Fig. 10. Response rate of different task load bound ϵ .

VI. CONCLUSION

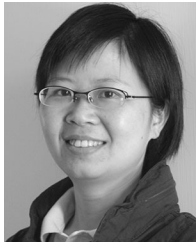
In this paper, we have proposed a cooperative task scheduling scheme for computation offloading in VC to improve the overall computing efficiency of the edge cloud. Considering the instability of resources, heterogeneity of computing capabilities, and inter-dependency of computing tasks in VC, it has been formulated as a task allocation problem, which is NP-hard. An MGA based method has been proposed to solve the problem. Extensive simulations have been conducted to demonstrate the efficiency of the proposed scheduling scheme. For future works, we will consider multiple types of services with specific requirements in VC, and more scenarios where different mobility models may apply.

REFERENCES

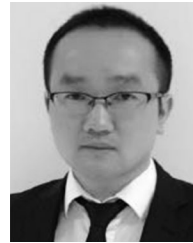
- [1] J. Ren, H. Guo, C. Xu, and Y. Zhang, "Serving at the edge: A scalable IOT architecture based on transparent computing," *IEEE Netw.*, vol. 31, no. 5, pp. 96–105, Aug. 2017.
- [2] T. G. Rodrigues, K. Suto, H. Nishiyama, and N. Kato, "Hybrid method for minimizing service delay in edge cloud computing through VM migration and transmission power control," *IEEE Trans. Comput.*, vol. 66, no. 5, pp. 810–819, May 2017.
- [3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 3, pp. 1628–1656, Mar. 2017.
- [4] J. A. Guerrero-ibanez, S. Zeadally, and J. Contreras-Castillo, "Integration challenges of intelligent transportation systems with connected vehicle, cloud computing, and internet of things technologies," *IEEE Wireless Commun.*, vol. 22, no. 6, pp. 122–128, Dec. 2015.
- [5] N. Cheng, N. Lu, N. Zhang, X. Shen, and J. W. Mark, "Vehicular wifi offloading: Challenges and solutions," *Veh. Commun.*, vol. 1, no. 1, pp. 13–21, Jan. 2014.
- [6] W. Quan, Y. Liu, H. Zhang, and S. Yu, "Enhancing crowd collaborations for software defined vehicular networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 80–86, Aug. 2017.
- [7] F. Lyu *et al.*, "Intelligent context-aware communication paradigm design for IOVs based on data analytics," *IEEE Netw.*, to be published.
- [8] Nvidia, "Nvidia DRIVE PX," 2018. [Online]. Available: <http://www.nvidia.ca/object/drive-px.html>
- [9] N. Cheng, N. Zhang, N. Lu, X. Shen, J. W. Mark, and F. Liu, "Opportunistic spectrum access for CR-VANETS: A game-theoretic approach," *IEEE Trans. Veh. Technol.*, vol. 63, no. 1, pp. 237–251, Jan. 2014.
- [10] N. Cheng *et al.*, "Big data driven vehicular networks," *IEEE Netw.*, pp. 1–8, Aug. 2018, doi: [10.1109/MNET.2018.1700460](https://doi.org/10.1109/MNET.2018.1700460).
- [11] 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.
- [12] L. Gu, D. Zeng, and S. Guo, "Vehicular cloud computing: A survey," in *Proc. IEEE GLOBECOM Workshops*, Atlanta, GA, USA, Dec. 2013, pp. 403–407.
- [13] F. Lyu *et al.*, "SS-MAC: A novel time slot-sharing MAC for safety messages broadcasting in vanets," *IEEE Trans. Veh. Technol.*, vol. 67, no. 4, pp. 3586–3597, Apr. 2018.
- [14] F. Ahmad, M. Kazim, A. Adnane, and A. Awad, "Vehicular cloud networks: Architecture, applications and security issues," in *Proc. IEEE/ACM UCC*, Limassol, Cyprus, Dec. 2015, pp. 571–576.
- [15] E. Lee, E. K. Lee, M. Gerla, and S. Y. Oh, "Vehicular cloud networking: Architecture and design principles," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 148–155, Feb. 2014.
- [16] I. Jang, S. Choo, M. Kim, S. Pack, and G. Dan, "The software-defined vehicular cloud: A new level of sharing the road," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 78–88, Jun. 2017.
- [17] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.
- [18] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
- [19] 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.
- [20] C. You, K. Huang, H. Chae, and B. H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
- [21] C. C. Lin, D. J. Deng, and C. C. Yao, "Resource allocation in vehicular cloud computing systems with heterogeneous vehicles and roadside units," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3692–3700, Oct. 2018.
- [22] N. Akhtar, S. C. Ergen, and O. Ozkasap, "Vehicle mobility and communication channel models for realistic and efficient highway vanet simulation," *IEEE Trans. Veh. Technol.*, vol. 64, no. 1, pp. 248–262, Jan. 2015.
- [23] K. A. Ali, O. Baala, and A. Caminada, "On the spatiotemporal traffic variation in vehicle mobility modeling," *IEEE Trans. Veh. Technol.*, vol. 64, no. 2, pp. 652–667, Feb. 2015.
- [24] C. Wang, Y. Li, D. Jin, and S. Chen, "On the serviceability of mobile vehicular cloudlets in a large-scale urban environment," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2960–2970, Oct. 2016.
- [25] H. Zhang, Q. Zhang, and X. Du, "Toward vehicle-assisted cloud computing for smartphones," *IEEE Trans. Veh. Technol.*, vol. 64, no. 12, pp. 5610–5618, Dec. 2015.
- [26] W. Zhang, Y. Wen, and D. O. Wu, "Collaborative task execution in mobile cloud computing under a stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 14, no. 1, pp. 81–93, Jan. 2015.
- [27] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.
- [28] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.
- [29] Y. Fang, "Hyper-Erlang distribution model and its application in wireless mobile networks," *Wireless Netw.*, vol. 7, no. 3, pp. 211–219, May 2001.
- [30] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, to be published, doi: [10.1109/TCC.2016.2560808](https://doi.org/10.1109/TCC.2016.2560808).
- [31] Z. Lu, J. Zhao, Y. Wu, and G. Cao, "Task allocation for mobile cloud computing in heterogeneous wireless networks," in *Proc. IEEE 24th Int. Conf. Comput. Commun. Netw.*, Las Vegas, NV, USA, Aug. 2015, pp. 1–9.



Fei Sun received the B.S. degree from the School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2014. He is currently working toward the Ph.D degree with the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include distributed caching in wireless networks, data and computation offloading, and vehicular ad hoc networks.



Fen Hou received the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2008. She is an Assistant Professor in the Department of Electrical and Computer Engineering with the University of Macau, Macau, China. She was a Postdoctoral Fellow with the Electrical and Computer Engineering with the University of Waterloo, from 2008 to 2009, and also with the Department of Information Engineering with the Chinese University of Hong Kong, Shatin, Hong Kong, from 2009 to 2011. She is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Macau. Her research interests include resource allocation and scheduling in broadband wireless networks, protocol design and QoS provisioning for multimedia communications in broadband wireless networks, mechanism design and optimal user behavior in mobile crowd sensing networks, and mobile data offloading. She was a recipient of the IEEE Globecom Best Paper Award in 2010 and the Distinguished Service Award in the IEEE MMTC in 2011. She served as the Co-Chair of the ICCS 2014 Special Session on Economic Theory and Communication Networks, the INFOCOM 2014 Workshop on Green Cognitive Communications and Computing Networks, the IEEE Globecom Workshop on Cloud Computing System, Networks, and Application 2013 and 2014, the ICCS 2015 Selected Topics in Communications Symposium, and the ICC 2016 Communication Software Services and Multimedia Application Symposium. She is currently serving as the Vice-Chair (Asia) of the IEEE ComSoc Multimedia Communications Technical Committee. She also serves as an Associate Editor for the IET Communications.



Haibo Zhou (M'14–SM'18) received the Ph.D. degree in information and communication engineering in 2014 from Shanghai Jiao Tong University, Shanghai, China. From 2014 to 2017, he worked as a Postdoctoral Fellow with the Broadband Communications Research Group, Department of Electrical and Computer Engineering, University of Waterloo. He is currently an Associate Professor with the School of Electronic Science and Engineering, Nanjing University, Nanjing, China. His research interests include resource management and protocol design in cognitive radio networks and vehicular networks.



Lin Gui (M'08) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2002. Since 2002, she has been with the Institute of Wireless Communication Technology, Shanghai Jiao Tong University, Shanghai, China, where she is currently a Professor. Her current research interests include HDTV and wireless communications.



Nan Cheng (S'12–M'16) received the B.E. and M.S. degrees from Tongji University, Shanghai, China, in 2009 and 2012, respectively, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2016. He is currently working as a Postdoctoral fellow with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada and the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada, under the supervision of Prof. Ben Liang and Prof. Sherman (Xuemin) Shen.

His current research include big data in vehicular networks and self-driving system. His research interests also include performance analysis, MAC, opportunistic communication, and application of AI for vehicular networks.



Xuemin Shen (M'97–SM'02–F'09) received the B.Sc. degree from Dalian Maritime University, Dalian, China, in 1982, and the M.Sc. and Ph.D. degrees from Rutgers University, New Brunswick, NJ, USA, in 1987 and 1990, respectively, all in electrical engineering. He is an University Professor and the Associate Chair for Graduate Studies, Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include resource management, wireless network security, social networks, smart grid, and vehicular

ad hoc and sensor networks. He served as the Technical Program Committee Chair/Co-Chair for the IEEE Globecom'16, Infocom'14, IEEE VTC'10 Fall, and Globecom'07, the Symposia Chair for the IEEE ICC'10, the Tutorial Chair for the IEEE VTC'11 Spring and the IEEE ICC'08, the General Co-Chair for ACM Mobihoc'15, Chinacom'07, and the Chair for the IEEE Communications Society Technical Committee on Wireless Communications. He also serves/served as the Editor-in-Chief for the IEEE INTERNET OF THINGS JOURNAL, IEEE NETWORK, *Peer-to-Peer Networking and Application*, and *IET Communications*; a Founding Area Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS; an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, *Computer Networks*, *ACM/Wireless Networks*, etc.; and the Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE WIRELESS COMMUNICATIONS, the IEEE COMMUNICATIONS MAGAZINE, etc. He was the recipient of the Excellent Graduate Supervision Award in 2006, and the Premiers Research Excellence Award in 2003 from the Province of Ontario, Canada. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.



Miao Wang received the B.Sc. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2007, the M.Sc. degree from Beihang University, Beijing, China, in 2010, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2015. She is an Assistant Professor with the Department of Electrical and Computer Engineering, Miami University, Oxford, OH, USA. Her current research interests include electric vehicles charging/discharging strategy design in smart grids,

traffic control, capacity and delay analysis, and routing protocol design for vehicular networks.