Self-Learning Based Computation Offloading for Internet of Vehicles: Model and Algorithm

Quyuan Luo, Changle Li, *Senior Member, IEEE*, Tom H. Luan, *Senior Member, IEEE*, Weisong Shi, *Fellow, IEEE*, and Weigang Wu, *Member, IEEE*

***Abstract*— With the fast development of Internet of Vehicles (IoV), various types of computation-intensive vehicular applications pose significant challenges to resource-constrained vehicles. The emerging Vehicular Edge Computing (VEC) and Edge Intelligence (EI) can alleviate this situation by offloading the computation tasks of vehicles to the roadside edge servers. However, with many vehicles contending for the communication and computation resources at the same time, how to quickly and efficiently make an optimal computation offloading decision for individual vehicles represents a fundamental research issue. In this paper, we propose a self-learning based distributed computation offloading scheme for IoV. Note that without any centralized controller, a fully distributed algorithm is necessary. The proposed scheme is devised based on a game-theoretic model. Specifically, through establishing an offloading framework with communication and computation for IoV, the computation offloading problem is first formulated as a distributed offloading decision-making game, in which each vehicle as a player makes its best response decision to minimize its joint cost (including latency and offloading cost). The existence of Nash Equilibrium can be proved. We then propose a self-learning based distributed computation offloading (`DISCO`) algorithm to reach the Nash Equilibrium, where a mutually satisfactory solution among vehicles is obtained and no vehicle is willing to change its decision. Using extensive simulations, we verify that `DISCO` can outperform the counterparts and achieve at least an order-of-magnitude improvement on time overhead and 88% performance gain on message overhead, only at up to 12% performance loss on joint cost over the centralized scheme.**

Manuscript received July 31, 2020; revised January 3, 2021 and March 10, 2021; accepted March 21, 2021. Date of publication April 13, 2021; date of current version September 10, 2021. This work was supported in part by the Fundamental Research Funds for the Central Universities, in part by the National Natural Science Foundation of China under Grant U1801266 and Grant 61731017, and in part by the Scholarship from the China Scholarship Council. The associate editor coordinating the review of this article and approving it for publication was Y. Shen. *(Corresponding author: Changle Li.)*

Quyuan Luo is with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China, also with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and also with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: qyluo@swjtu.edu.cn).

Changle Li is with the State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, China, and also with the Research Institute of Smart Transportation, Xidian University, Xi'an 710071, China (e-mail: clli@mail.xidian.edu.cn).

Tom H. Luan is with the School of Cyber Engineering, Xidian University, Xi'an 710071, China (e-mail: tom.luan@xidian.edu.cn).

Weisong Shi is with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: weisong@wayne.edu).

Weigang Wu is with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China (e-mail: wuweig@mail.sysu.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TWC.2021.3071248.

Digital Object Identifier 10.1109/TWC.2021.3071248

***Index Terms*— Vehicular edge computing, distributed computation offloading, game theory, Nash equilibrium, self-learning.**

I. Introduction

THE Internet of Vehicles (IoV) has recently attracted increasing interests from both academic and industry [1]–[3]. By integrating the advanced computation and communication in one platform, IoV can support various types of vehicular applications, such as autonomous driving, precise fleet management, and real-time video analytics [4], [5], and plays a crucial role toward the next-generation intelligent transportation system [6].

The powerful and resource-hungry applications of IoV, however, require intensive real-time computation, which poses significant challenges on resource-constrained IoV [7]. To address the issue, the paradigm of Vehicular Edge Computing (VEC) has been proposed [8]. In VEC, the IoV can offload the computation-intensive tasks to roadside units (RSUs) that are equipped with edge servers [9] which deploys and trains powerful machine learning models, i.e., Edge Intelligence (EI), to help data processing for driving through IoV [10]–[12]. However, note that each connected IoV to VEC may present random demand and different urgencies of tasks, how to make proper offloading decisions for IoV by jointly considering radio and computing resources deserves investigation.

Several previous efforts have focused on the computation offloading [9], [13]–[18]. In these works, the computation offloading is mostly formulated as a resource allocation problem through either minimizing the total latency or cost or maximizing the system utility. Marvelous solutions are proposed to solve these optimization problems. They often adopted centralized optimization methods by collecting complete information from vehicles, which requires frequent state information updating to optimize the system performance and results in a high system overhead [19]. Even so, global optimal solutions generally cannot be obtained due to the complexity of the centralized optimization problems, instead, some kind of heuristic solutions are obtained. Moreover, some vehicles may be unwilling to send their information due to privacy concerns and hence are unwilling to participate in centralized optimization. Accordingly, it is imperative for vehicles to *self-learn* the best offloading decisions for their own profits in a distributed manner.

Motivated by the aforementioned discussion, we aim to propose and design a distributed computation offloading scheme

for IoV, where each vehicle can make an offloading decision independently. To this end, in this paper, we propose a self-learning based distributed computation offloading scheme for IoV. Specifically, we first establish an offloading framework with communication and computation for IoV, in which each IoV tries to obtain its best computation offloading decision in pursuit of a minimum joint cost. Then, we formulate the computation offloading problem as a distributed offloading decision-making game, which can analyze the intersections among multiple IoVs that act in their own interests. By utilizing the concept of *best response* [20] and *potential game* [21], we prove the existence of Nash Equilibrium [22] of our formulated game. To reach the Nash Equilibrium, we propose a self-learning based distributed computation offloading (DISCO) algorithm. With DISCO, each IoV can learn the best response decision automatically and independently towards an equilibrium state, where no IoV is willing to unilaterally change its offloading decision. The contributions of this paper can be summarized as follows.

1) *Model*: We establish an offloading framework with both communication and computation for IoV. Under the framework, we formulate the computation offloading problem as a distributed offloading decision-making game to analyze the intersections among multiple IoV, where each IoV acts in its own interest.
2) *Algorithm*: After we prove the existence of Nash Equilibrium of the game, we propose a self-learning based distributed computation offloading (DISCO) algorithm, where each IoV learns from its individual information and the decision profile of other vehicles to make its best response decision. No control center is required, DISCO is a very practical algorithm.
3) *Validation*: We use real-world vehicular traces to conduct extensive simulations, which demonstrates the effectiveness of our proposed DISCO over counterparts. How different variables (such as task data size, number of vehicles, communication resource, weightings of latency and cost) impact the system performance is also presented by simulation results.

The remainder of this paper is organized as follows. The related work is presented in Section II. In Section III, we depict the offloading framework with communication and computation for IoV. The game formulation and the existence of Nash Equilibrium are introduced in Section IV. The proposed DISCO algorithm is presented in Section V. In Section VI, extensive simulation results are discussed. The conclusion is drawn in Section VII.

II. Related Work

In this section, we survey the existing literature on computation offloading for IoV both in centralized and decentralized manners.

There are some works studying computation offloading. The authors in [14] propose a dual-side dynamic joint task offloading and resource allocation algorithm in vehicular networks (DDORV), which utilizes Lyapunov optimization theory to minimize the averaged cost of mobile edge computing (MEC) enabled roadside unites and vehicular terminal. Through derivation and comparing the values of local processing cost and task offloading cost, the optimization problem on the vehicular terminal side is solved. For the optimization issue on the MEC server side, the Lagrangian dual decomposition and continuous relaxation method are adopted. The authors in [9] propose a cloud-based MEC offloading framework in vehicular networks, where both the heterogeneous requirements of the mobility of the vehicles and the computation tasks are considered. Based on the analysis of the characteristics of various offloading strategies, the authors further propose a predictive-mode transmission scheme for task-file uploading. The work in [23] proposes a Lyapunov-based dynamic offloading decision algorithm for flexible subtasks by jointly considering energy consumption and packet drop rate. To address the overload problem in the edge server, the work in [8] integrates load balancing and offloading problems in the VEC network.

Most of the existing computation offloading problems are mixed-integer non-linear programming (MINLP) problems. They are generally NP-hard and are not computable in polynomial time with existing general solvers. Generally speaking, the complex optimization problem is decomposed into subproblems, and the near-optimal solution is derived by solving those subproblems respectively [14]. Recently, many works consider utilizing deep reinforcement learning based methods to solve the complex computation offloading problem. The authors in [24] present a reinforcement learning based mobile offloading scheme for edge computing against jamming attacks and interference, which uses safe reinforcement learning to avoid choosing the risky offloading policy that fails to meet the computational latency requirements of the tasks. The authors in [25] propose two deep reinforcement learning based dynamic computation offloading algorithms for mobile edge computing systems with energy harvesting devices, which addressed the challenges of continuous-discrete hybrid action spaces and coordination among devices.

Other works study the computation offloading in a decentralized manner. The literature [16] proposes a hierarchical cloud-based VEC offloading framework to improve the quality of offloading service that may be endangered by the computation limitation of MEC servers. And a Stackelberg game-based method is adopted to maximize the utilities of both computing servers and vehicles. Authors in [26] consider the latency of computation offloading as the computation overhead of vehicles and try to reduce it through a multi-user distributed computation offloading algorithm based on game theory. Authors in [27] and [28] propose a game theory-based multi-user distributed computation offloading algorithms for mobile cloud computing and mobile-edge cloud computing, respectively. The work in [29] proposes a fully distributed computation offloading (FDCO) algorithm to address the multi-user computation offloading problem for cloudlet-based mobile cloud computing in a multi-channel wireless contention environment. The work in [30] considers the problem of computation offloading while achieving a trade-off between execution time and energy consumption in an unmanned areal vehicle (UAV) network, where the combination of energy overhead and latency is minimized by

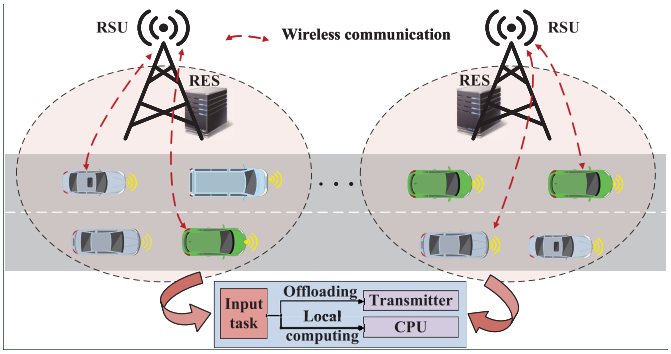


Fig. 1. Computation offloading in a VEC network.

the designed game theory model. The work in [31] formulate a distributed computation offloading problem among mobile users as an exact potential game and propose a distributed offloading scheme based on Q-learning and better-response. To improve the task offloading delay performance, authors in [19] and [32] respectively proposed an adaptive learning based offloading algorithm and an online learning-based task replication algorithm based on multi-armed bandit theory. The authors in [33] propose a contract-based traffic offloading and resource allocation mechanism for the software-defined wireless network (SDWN)-cased heterogeneous ultra-dense networks (HetUDN), where each small-cell base station (SBS) selects the contract that achieves its own maximum utility.

All these works above are marvelous solutions. However, for the centralized computation offloading, complete information of all vehicles should be collected, which results in a high system overhead. Generally speaking, the global optimal solution cannot be obtained even with the complete information because of the complexity of centralized methods. For the distributed computation offloading, most existing works only consider a single performance index such as delay or a simple combination of delay and energy consumption, ignoring other important performance indexes such as communication cost and computation cost. In light of the existing works, considering energy consumption, communication cost and computation cost as the offloading cost, we formulate the computation offloading as a distributed game to minimize the combination of latency and offloading cost and propose a self-learning based distributed computation offloading (DISCO) algorithm to reach the Nash Equilibrium, as well as provide detailed analysis on how different variables impact the system performance from simulation results.

III. Offloading Framework With Communication and Computation for IoV

A. System Description

For convenience, the main notations used are summarized in Table I. Fig. 1 shows the computation offloading in a VEC network. The road is divided into segments, and each covered by a roadside unit (RSU) with a roadside edge server (RES). We consider an LTE-V network composed of vehicles and roadside units (RSUs) deployed along the road. And each vehicle has an LTE-V radio interface to establish

TABLE I
Major Notations

Notation	Explanation
B	Bandwidth of each licensed channel
C_n	Processing density of computation task T_n
D_n	Data size of computation task T_n
d_n	Distance between vehicle n and RSU
f^e	Processing capability of RSU
f^l	Processing capability of vehicle n
h	Channel fading coefficient
κ_n	Coefficients related to power in vehicle
κ_e	Coefficients related to power in RSU
L	Number of licensed channels for V2I communication
M	Number of processing cores of RES enable RSU
$\mathcal{N}, N$	Set and number of vehicles
P_n^{tr}	Transmission power of vehicle n
χ_n	Offloading decision of vehicle n
$\boldsymbol{\chi}$	Offloading decision profile of all vehicles
$\boldsymbol{\chi}_{-n}$	Offloading decisions of other vehicles
ϑ	Path loss exponent
ω_0	White Gaussian noise power
λ_1, λ_2	Weighting parameters of latency and cost
ϱ	Energy consumption cost of one unit energy during task computing and transmitting
ξ	Communication cost to transmit one unit of task data by using licensed V2I channels
γ	Computing cost to execute one CPU cycle

a communication link with RSU. We consider a coverage area of one RSU and a set of $\mathcal{N} = \{1, 2, \ldots, N\}$ Internet of Vehicles (hereinafter referred to as *vehicle* for short). The RSU can provide powerful computing capacity due to the deployed RES. Each vehicle has a latency-sensitive and computation-intensive task to be processed. We use two items to describe the computation task of vehicle n ($n \in \mathcal{N}$), i.e., $T_n \triangleq \{D_n, C_n\}$, where D_n stands for the data size of T_n, and C_n stands for the processing density (in CPU cycles/bit) of T_n. Vehicles can establish a communication link with RSU through the orthogonal licensed vehicle-to-infrastructure (V2I) channels, each with the bandwidth of B. The number of licensed channels for V2I communication is denoted by L. We use an indicator χ_n to denote the task offloading decision of vehicle n, where $\chi_n = 0$ indicates that task T_n is computed locally by vehicle n, and $\chi_n = 1$ indicates that task T_n is offloaded to RSU to be processed through V2I communication. In the following, we will elaborate on the two cases, respectively.

B. Task Processed Locally

If vehicle n choose $\chi_n = 0$, task T_n will be processed locally by vehicle n. Let f_n^l denote the processing capability (i.e., the amount of CPU frequency in cycles/s) at vehicle n assigned for local computing, then the power consumption for

vehicle n to process task T_n locally is expressed as

$$p_n^l = \kappa_n (f_n^l)^3, \tag{1}$$

where κ_n is a coefficient related to power in vehicle n [34]. The local execution time of task T_n is then given by

$$t_n^l = \frac{D_n C_n}{f_n^l}. \tag{2}$$

Accordingly, the energy consumption of vehicle n for local processing is expressed as

$$E_n^l = p_n^l t_n^l = \kappa_n D_n C_n (f_n^l)^2. \tag{3}$$

C. Task Offloaded to RSU

If vehicle n choose $\chi_n = 1$, task T_n will be offloaded to RSU to be processed. Since the processing result is usually very tiny, we neglect the output return process and just focus on transmitting data to RSU [14]. There exist two procedures to accomplish the task computing in RSU, which will be presented in the following.

1) Task Transmission: For ease of analysis, we consider the system to be quasi-static so that the wireless channels and the topology of the system keep unchanged during an offloading period [27]. Let h denote the channel fading coefficient, which is modeled as a circularly symmetric complex Gaussian random variable [35]. When task is transmitted from vehicle n to RSU on licensed V2I channels, the transmission rate is given by

$$r_n(\boldsymbol{\chi}) = \frac{L}{\sum_{n\in\mathcal{N}} \chi_n} B \log_2\left(1 + \frac{P_n |h|^2}{\omega_0 (d_n)^{\vartheta}}\right), \tag{4}$$

where P_n is the transmission power of vehicle n, ω_0 denotes the white Gaussian noise power, $\boldsymbol{\chi}$ is the decision profile of all vehicles, denoted by $\boldsymbol{\chi} = (\chi_1, \chi_2, \ldots, \chi_N)$, d_n and ϑ denote the distance from vehicle n to RSU and the path loss exponent, respectively. Since we consider all $\sum_{n\in\mathcal{N}} \chi_n$ vehicles share the L channels, each vehicle can obtain $\frac{L}{\sum_{n\in\mathcal{N}} \chi_n}$ channels. It is notable that the powerful RES renders the edge computing time of offloaded tasks rather small. Therefore, for simplicity, the vehicle is assumed to stay stationary while performing edge computing [36]. Accordingly, when transmitting task T_n, the transmission time is expressed as

$$t_n^{tr}(\boldsymbol{\chi}) = \frac{D_n}{r_n(\boldsymbol{\chi})}, \tag{5}$$

and the energy consumption is expressed as

$$E_n^{tr}(\boldsymbol{\chi}) = P_n t_n^{tr} = \frac{P_n D_n}{r_n(\boldsymbol{\chi})}. \tag{6}$$

2) Task Computed by RSU: After task T_n is transmitted from vehicle n to RSU, it will be computed by the deployed RES. Let M denote the number of processing cores of the RES enabled RSU, and the processing capability (i.e., the amount of CPU frequency in cycles/s) of each core as f^e, then the power consumption of each core to compute task is expressed as

$$p^e = \kappa_e (f^e)^3, \tag{7}$$

where κ_e is a coefficient related to power in RSU [34]. Accordingly, the execution time and energy consumption of RSU for compute task T_n can be expressed as

$$t_n^e(\boldsymbol{\chi}) = \frac{D_n C_n}{\frac{M}{\sum_{n\in\mathcal{N}} \chi_n} f^e} = \frac{D_n C_n \sum_{n\in\mathcal{N}} \chi_n}{M f^e}, \tag{8}$$

and

$$E_n^e(\boldsymbol{\chi}) = \frac{M}{\sum_{n\in\mathcal{N}} \chi_n} p^e t_n^e(\boldsymbol{\chi}) = \kappa_e D_n C_n (f^e)^2. \tag{9}$$

Accordingly, the total latency for processing task T_n when it is offloaded to RSU is expressed as

$$\begin{aligned} t_n^{off}(\boldsymbol{\chi}) &= t_n^{tr}(\boldsymbol{\chi}) + t_n^e(\boldsymbol{\chi}) \\ &= \frac{D_n}{r_n(\boldsymbol{\chi})} + \frac{D_n C_n \sum_{n\in\mathcal{N}} \chi_n}{M f^e}. \end{aligned} \tag{10}$$

Similarly, the total energy consumption for processing task T_n when it is offloaded to RSU is expressed as

$$\begin{aligned} E^{off}(\boldsymbol{\chi}) &= E_n^{tr}(\boldsymbol{\chi}) + E_n^e(\boldsymbol{\chi}) \\ &= \frac{P_n D_n}{r_n(\boldsymbol{\chi})} + \kappa_e D_n C_n (f^e)^2. \end{aligned} \tag{11}$$

D. Joint Cost and Problem Formulation

For a given task T_n, costs for processing this task would be produced. Just like the cost defined in [27], the execution cost defined in [29], and the utility function defined in [30], we use the term *joint cost* to define the overall cost to process task T_n as the combination of latency and offloading cost, expressed as

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) = \lambda_1 \boldsymbol{Latency} + \lambda_2 \boldsymbol{Cost}, \tag{12}$$

where $\boldsymbol{\chi}_{-n} = (\chi_1, \ldots, \chi_{n-1}, \chi_{n+1}, \chi_N)$ denotes the task offloading decisions of other vehicles, λ_1 and λ_2 denote the weighting parameters of latency and offloading cost, respectively.[1] Specifically, the $\boldsymbol{Cost}$ when $\chi_n = 0$ only includes the energy consumption for vehicle n computing task T_n, while the $\boldsymbol{Cost}$ when $\chi_n = 1$ includes three aspects: *a)* the energy consumption of transmitting and computing task T_n; *b)* the communication cost for using licensed V2I channels; and *c)* the computing cost for RSU processing task T_n. Accordingly, the costs for $\chi = 0$ and $\chi = 1$ can be formulated as

$$O_n^l = \varrho E_n^l, \tag{13}$$

and

$$O_n^{off}(\boldsymbol{\chi}) = \varrho\Big(E_n^{tr}(\boldsymbol{\chi}) + E_n^e(\boldsymbol{\chi})\Big) + \xi D_n + \gamma D_n C_n, \tag{14}$$

respectively, where ϱ is a weighting coefficient indicating the energy consumption cost of one unit energy during task computing and transmitting [38], ξ is a coefficient indicating the communication cost required to transmit one unit of task data by using licensed V2I channels, γ is a coefficient indicating the computing cost to execute one CPU cycle. Based on the analysis above, the joint cost of an arbitrary vehicle n is then given by

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) = \begin{cases} \lambda_1\ t_n^l + \lambda_2\ O_n^l, & \text{if } \chi_n = 0 \\ \lambda_1 t_n^{off}(\boldsymbol{\chi}) + \lambda_2\ O_n^{off}(\boldsymbol{\chi}), & \text{if } \chi_n = 1 \end{cases} \tag{15}$$

[1]The parameters can be adjusted according to different requirements for latency and offloading cost [30] and can be also determined based on the multiple criteria decision making theory [37].

It is noting that the value of $\lambda_1 t_n^{off}(\boldsymbol{\chi}) + \lambda_2\ O_n^{off}(\boldsymbol{\chi})$ is related to the value of $\sum_{n\in\mathcal{N}} \chi_n$. Since different vehicles may have different tasks, different amount of data will be transmitted to the RSU. When one vehicle finishes its transmission while others do not, or a task is finished by the RSU while others is not, more communication or computation resources will be freed up. If this part of resources are utilized by other vehicles or tasks, the value of $\lambda_1 t_n^{off}(\boldsymbol{\chi}) + \lambda_2\ O_n^{off}(\boldsymbol{\chi})$ will decreased according to formulas (4), (5), (6), (10) and (11). For simplicity, in this paper, we assume that the communication and computation resources assigned to each vehicle keep fixed once the optimal decisions are made.

For a given vehicle n, it's purpose is to minimize its joint cost during the computation offloading decision process, which is expressed as

$$\min_{\chi_n\in\{0,1\}} U_n(\chi_n, \boldsymbol{\chi}_{-n}), \quad \forall n \in \mathcal{N}. \tag{16}$$

It is obvious from the models presented above that vehicles' decisions $\boldsymbol{\chi}$ are coupled. This is because the decision made by any vehicle would influence other vehicles in the VEC network. For example, if too many vehicles make the same offloading decision through V2I communications, they may get less licensed channels and this would lead to a low data rate according to formula (4). A low data rate would therefore lead to a higher latency and more energy consumption during transmission. Moreover, in this case, less computation resource would be allocated to each task, which would lead to a higher computing time according to formula (8). Instead, it would be more beneficial for some vehicles to choose the local computing decision. To obtain the best computation offloading decision towards a minimum joint cost defined in formula (12) among vehicles, we design and implement a game theory based method in the following.

IV. Game Formulation and Nash Equilibrium

Different vehicles may pursue their own interests and have different requirements for latency and offloading cost, the centralized methods usually have a very high complexity due to a large state space involving many vehicles and tasks, which results in a longer convergence time to obtain optimal offloading decisions [32]. By taking full advantage of the intelligence of individual vehicles, game theory is a powerful framework to analyze the intersections among vehicles that act in their own interests with low complexity [39]. Under the game theory-based framework, vehicles can self-learn the best offloading decisions and self-organize into a Nash Equilibrium state [22]. In such a state, no vehicle is willing to change its offloading decision. Thus, we believe that a game theory-based method can ease the burden of leveraging a complex centralized computation offloading method and reduce the controlling and signaling overhead between RSU and vehicles.

A. Game Formulation

According to formula (16), each vehicle tries to minimize its joint cost $U_n(\chi_n, \boldsymbol{\chi_{-n}})$. We consider the computation offloading problem within an offloading period. The distributed computation offloading game is formulated as

$$\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{U} \rangle, \tag{17}$$

which consists of three parts:

- $\mathcal{N}$ is the set of players (i.e., vehicles);
- $\mathcal{A} = \{\mathcal{A}_n\}_{n\in\mathcal{N}}$ is the set of decision space of all players, where $\mathcal{A}_n = \{0, 1\}$ denotes the set of actions player n can take;
- $\mathcal{U} = \{U_1(\chi_1, \boldsymbol{\chi}_{-1}), U_2(\chi_2, \boldsymbol{\chi}_{-2}), \ldots, U_N(\chi_N, \boldsymbol{\chi}_{-N})\}$ is the set of joint cost of all players, where $U_n(\chi_n, \boldsymbol{\chi}_{-n})$ denotes the joint cost of vehicle n, as defined in formula (15).

Each player adjust its decision based on formula (15) to minimize its joint cost. For example, for two possible decisions $\dot{\chi}_n$ and $\ddot{\chi}_n$ of player n, if $U_n(\dot{\chi}_n, \boldsymbol{\chi}_{-n}) < U_n(\ddot{\chi}_n, \boldsymbol{\chi}_{-n})$, which means decision $\dot{\chi}_n$ is more profitable than $\ddot{\chi}_n$ for player n, then player n would independently and selfishly choose decision $\dot{\chi}_n$ to reduce its joint cost. Otherwise, $\ddot{\chi}_n$ would be a prefer of player n. Due to the independence of players in making offloading decisions, we call the formulated game $\mathcal{G}$ as the distributed computation offloading game. In the following, we introduce Nash Equilibrium [20], a very important concept in game theory.

Definition 1 (Nash Equilibrium): For game $\mathcal{G}$, we call $\boldsymbol{\chi}^* = (\chi_1^*, \chi_2^*, \ldots, \chi_N^*)$ a Nash Equilibrium if and only if no vehicle can further improve its profit by unilaterally changing its decision at the equilibrium $\boldsymbol{\chi}^*$, i.e.,

$$U_n(\chi_n^*, \boldsymbol{\chi}_{-n}^*) < U_n(\chi_n, \boldsymbol{\chi}_{-n}^*), \quad \forall \chi_n \in \mathcal{A}_n, \forall n \in \mathcal{N}. \tag{18}$$

The Nash Equilibrium has a very important property, i.e., the self-stability property that when at the equilibrium $\boldsymbol{\chi}^*$, each player obtains its best decision and would not change it. To reach such an equilibrium state, we now need to prove the existence of Nash Equilibrium.

B. Existence of Nash Equilibrium of $\mathcal{G}$

To prove the existence of Nash Equilibrium of $\mathcal{G}$, we first introduce *best response* [20].

Definition 2 (Best Response): Given the computation offloading decisions $\boldsymbol{\chi}_{-n}$, if the computation offloading decision $\chi_n^* \in \mathcal{A}_n$ meets:

$$U_n(\chi_n^*, \boldsymbol{\chi}_{-n}) < U_n(\chi_n, \boldsymbol{\chi}_{-n}), \quad \forall \chi_n \in \mathcal{A}_n, \tag{19}$$

we call χ_n^* a best response.

After observing formulas (18) and (19), it is obvious that all vehicles make their best response decisions at the Nash Equilibrium $\boldsymbol{\chi}^*$. In the following, we introduce how to make the best response.

Lemma 1: Given the computation offloading decisions $\boldsymbol{\chi}_{-n}$ of other vehicles, vehicle n makes its best response according to the following formula:

$$\chi_n^* = \begin{cases} 1, & \sum_{i\in\mathcal{N}\setminus\{n\}} \chi_i < \Gamma_n - 1, \\ 0, & \textit{otherwise}, \end{cases} \tag{20}$$

where Γ_n *is expressed as*

$$\Gamma_n = \frac{\lambda_1 t_n^l + \lambda_2 \varrho E_n^l - \Psi_n}{\left(\frac{\lambda_1 D_n + \lambda_2 \varrho P_n D_n}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0 (d_n)^{\vartheta}})} + \frac{\lambda_1 D_n C_n}{Mf^e}\right)}, \tag{21}$$

and Ψ_n *is expressed as*

$$\Psi_n = \lambda_2(\varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n). \tag{22}$$

Proof: According to formula (15), the joint cost can be reformulated as

$$\begin{aligned} &U_n(\chi_n, \boldsymbol{\chi}_{-n}) \\ &= (1-\chi_n)(\lambda_1\ t_n^l + \lambda_2\ O_n^l) \\ &\quad + \chi_n(\lambda_1\ t_n^{off}(\boldsymbol{\chi}) + \lambda_2\ O_n^{off}(\boldsymbol{\chi})) \\ &= (1-\chi_n)\Big(\lambda_1\ t_n^l + \lambda_2 \varrho E_n^l\Big) \\ &\quad + \chi_n\Big(\lambda_1\Big(\frac{D_n}{r_n(\boldsymbol{\chi})} + \frac{D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e}\Big) \\ &\quad + \lambda_2\Big(\frac{\varrho P_n D_n}{r_n(\boldsymbol{\chi})} + \varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n\Big)\Big). \end{aligned} \tag{23}$$

If the best response decision of vehicle n is $\chi_n^* = 1$, according to Definition 2, we have

$$U_n(1, \boldsymbol{\chi}_{-n}) < U_n(0, \boldsymbol{\chi}_{-n}). \tag{24}$$

Combining formulas (23) and (24), we have

$$\begin{aligned} &\lambda_1\Big(\frac{D_n}{r_n(\boldsymbol{\chi})} + \frac{D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e}\Big) \\ &\quad + \lambda_2\Big(\frac{\varrho P_n D_n}{r_n(\boldsymbol{\chi})} + \varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n\Big) \\ &< \lambda_1 t_n^l + \lambda_2 \varrho E_n^l. \end{aligned} \tag{25}$$

That is,

$$\begin{aligned} &\frac{1}{r_n(\boldsymbol{\chi})}(\lambda_1 D_n + \lambda_2 \varrho P_n D_n) \\ &< \lambda_1 t_n^l + \lambda_2 \varrho E_n^l - \lambda_2(\varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n) \\ &\quad - \frac{\lambda_1 D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e}. \end{aligned} \tag{26}$$

For the sake of simplicity, we use Ψ to replace $\lambda_2(\varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n)$, and let's substitute formula (4) into formula (26), we have

$$\begin{aligned} &\frac{\sum_{i\in\mathcal{N}}\chi_i}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0(d_n)^{\vartheta}})}(\lambda_1 D_n + \lambda_2 \varrho P_n D_n) \\ &\quad < \lambda_1 t_n^l + \lambda_2 \varrho E_n^l - \Psi - \frac{\lambda_1 D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e}. \end{aligned} \tag{27}$$

That is,

$$\begin{aligned} &\sum_{i\in\mathcal{N}}\chi_i\Big(\frac{\lambda_1 D_n + \lambda_2 \varrho P_n D_n}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0(d_n)^{\vartheta}})} + \frac{\lambda_1 D_n C_n}{Mf^e}\Big) \\ &\quad < \lambda_1 t_n^l + \lambda_2 \varrho E_n^l - \Psi. \end{aligned} \tag{28}$$

Then we have

$$\sum_{i\in\mathcal{N}}\chi_i < \Gamma_n \triangleq \frac{\lambda_1 t_n^l + \lambda_2 \varrho E_n^l - \Psi}{\left(\frac{\lambda_1 D_n + \lambda_2 \varrho P_n D_n}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0(d_n)^{\vartheta}})} + \frac{\lambda_1 D_n C_n}{Mf^e}\right)}. \tag{29}$$

That is,

$$\sum_{i\in\mathcal{N}\setminus\{n\}}\chi_i < \Gamma_n - 1. \tag{30}$$

■

Now, we need to prove that the formulated game $\mathcal{G}$ has a Nash Equilibrium, and thus eventually converges after each vehicle makes the best response decision iteratively. To this end, we resort to the concept of *potential game* [21].

Definition 3 (Potential Game): A game is called a potential game if there exists a function $P : \boldsymbol{\chi} = (\chi_1, \chi_2, \ldots, \chi_N) \to \mathbb{R}$ *such that* $\forall n \in \mathcal{N}$, $\forall \boldsymbol{\chi}_{-n} \in \Pi_{i\neq n}\mathcal{A}_i$, $\forall \chi_n, \chi_n' \in \mathcal{A}_n$,

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) - U_n(\chi_n', \boldsymbol{\chi}_{-n}) = P(\chi_n, \boldsymbol{\chi}_{-n}) - P(\chi_n', \boldsymbol{\chi}_{-n}). \tag{31}$$

Function P *is a potential for the potential game.*

Since the potential game has the *finite improvement property* (FIP) that any better response updating process must be finite and lead to a Nash Equilibrium [21], [27], we now only need to prove that our formulated game $\mathcal{G}$ is a potential game.

Theorem 1: Our formulated distributed computation offloading game $\mathcal{G} = \langle \mathcal{N}, \mathcal{A}, \mathcal{U} \rangle$ *is a potential game with a potential function and hence always has the FIP and a Nash Equilibrium.*

Proof: According to formulas (15) and (23), we reformulate $U_n(\chi_n, \boldsymbol{\chi}_{-n})$ as

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) = \begin{cases} W_n^l, & \text{if } \chi_n = 0, \\ W_n^{off}(s), & \text{if } \chi_n = 1, \end{cases} \tag{32}$$

where

$$W_n^l = \lambda_1\ t_n^l + \lambda_2 \varrho E_n^l, \tag{33}$$

$$\begin{aligned} W_n^{off}(s) &= \lambda_1\Big(\frac{D_n}{r_n(\boldsymbol{\chi})} + \frac{D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e}\Big) \\ &\quad + \lambda_2\Big(\frac{\varrho P_n D_n}{r_n(\boldsymbol{\chi})} + \varrho E_n^e(\boldsymbol{\chi}) + \xi D_n + \gamma D_n C_n\Big) \\ &= \frac{\lambda_1 D_n + \lambda_2 \varrho P_n D_n}{r_n(\boldsymbol{\chi})} + \frac{\lambda_1 D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e} + \Psi_n \\ &= \frac{\sum_{i\in\mathcal{N}}\chi_i}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0(d_n)^{\vartheta}})}(\lambda_1 D_n + \lambda_2 \varrho P_n D_n) \\ &\quad + \frac{\lambda_1 D_n C_n \sum_{i\in\mathcal{N}}\chi_i}{Mf^e} + \Psi_n \\ &= s\Omega_n + \Psi_n, \end{aligned} \tag{34}$$

where $s = \sum_{i\in\mathcal{N}}\chi_i$, Ω_n and Ψ_n are two replacement variables to replace some parameters in formula (34) for the sake of simplicity, and Ω_n is formulated as

$$\Omega_n = \frac{\lambda_1 D_n + \lambda_2 \varrho P_n D_n}{LB\log_2(1+\frac{P_n|h|^2}{\omega_0(d_n)^{\vartheta}})} + \frac{\lambda_1 D_n C_n}{Mf^e}, \tag{35}$$

and Ψ_n is as shown in formula (22).

Now we define a function $P : \boldsymbol{\chi} = (\chi_1, \chi_2, \ldots, \chi_N) \to \mathbb{R}$ as

$$P(\chi_n, \boldsymbol{\chi}_{-n}) = \sum_{v=1}^{s} W_n^{off}(v) + W_n^l \sum_{i=1}^{N} I_{\{a_i=0\}}, \tag{36}$$

where $I_{\{\cdot\}} = 1$ when "$\cdot$" is true and $I_{\{\cdot\}} = 0$ when "$\cdot$" is false. We suppose an arbitrary vehicle n ($n \in \mathcal{N}$) changes its decision from χ_n to χ'_n. Since $\chi_n, \chi'_n \in \{0, 1\}$, there are two cases considered: *1)* $\chi_n = 0$ and $\chi'_n = 1$; *2)* $\chi_n = 1$ and $\chi'_n = 0$. We will elaborate on the two cases in the following.

1) $\boldsymbol{\chi_n = 0}$ *and* $\boldsymbol{\chi'_n = 1}$*:* In this case, we have

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) = \lambda_1 t_n^l + \lambda_2 \varrho E_n^l = W_n^l, \tag{37}$$

and

$$U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = (s+1)\Omega_n + \Psi_n = W_n^{off}(s+1). \tag{38}$$

According to formula (36), the defined function P can be expressed as

$$P(\chi_n, \boldsymbol{\chi}_{-n}) = \sum_{v=1}^{s} W_n^{off}(v) + W_n^l + W_n^l \sum_{i=1}^{N} I_{\{a_i=0, i\neq n\}}, \tag{39}$$

and

$$P(\chi'_n, \boldsymbol{\chi}_{-n}) = \sum_{v=1}^{s+1} W_n^{off}(v) + W_n^l \sum_{i=1}^{N} I_{\{a_i=0, i\neq n\}}. \tag{40}$$

Then,

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) - U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = W_n^l - W_n^{off}(s+1), \tag{41}$$

and

$$\begin{aligned} &P(\chi_n, \boldsymbol{\chi}_{-n}) - P(\chi'_n, \boldsymbol{\chi}_{-n}) \\ &= \sum_{v=1}^{s} W_n^{off}(v) + W_n^l - \sum_{v=1}^{s+1} W_n^{off}(v) \\ &= W_n^l - W_n^{off}(s+1). \end{aligned} \tag{42}$$

Accordingly,

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) - U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = P(\chi_n, \boldsymbol{\chi}_{-n}) - P(\chi'_n, \boldsymbol{\chi}_{-n}). \tag{43}$$

2) $\boldsymbol{\chi_n = 1}$ *and* $\boldsymbol{\chi'_n = 0}$*:* In this case, we have

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) = (s)\Omega_n + \Psi_n = W_n^{off}(s), \tag{44}$$

and

$$U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = \lambda_1 t_n^l + \lambda_2 \varrho E_n^l = W_n^l. \tag{45}$$

According to formula (36), the defined function P can be expressed as

$$P(\chi_n, \boldsymbol{\chi}_{-n}) = \sum_{v=1}^{s} W_n^{off}(v) + W_n^l \sum_{i=1}^{N} I_{\{a_i=0, i\neq n\}}, \tag{46}$$

and

$$P(\chi'_n, \boldsymbol{\chi}_{-n}) = \sum_{v=1}^{s-1} W_n^{off}(v) + W_n^l + W_n^l \sum_{i=1}^{N} I_{\{a_i=0, i\neq n\}}. \tag{47}$$

Then,

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) - U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = W_n^{off}(s) - W_n^l, \tag{48}$$

and

$$\begin{aligned} &P(\chi_n, \boldsymbol{\chi}_{-n}) - P(\chi'_n, \boldsymbol{\chi}_{-n}) \\ &= \sum_{v=1}^{s} W_n^{off}(v) - (\sum_{v=1}^{s-1} W_n^{off}(v) + W_n^l) \\ &= W_n^{off}(s) - W_n^l. \end{aligned} \tag{49}$$

Accordingly,

$$U_n(\chi_n, \boldsymbol{\chi}_{-n}) - U_n(\chi'_n, \boldsymbol{\chi}_{-n}) = P(\chi_n, \boldsymbol{\chi}_{-n}) - P(\chi'_n, \boldsymbol{\chi}_{-n}). \tag{50}$$

■

Theorem 1 and the above derivation proves that $\mathcal{G}$ is a potential game such that the existence of Nash Equilibrium is guaranteed. And based on the FIP of $\mathcal{G}$, we design a self-learning based distributed computation offloading (DISCO) algorithm to reach the Nash Equilibrium in the following.

V. Self-Learning Based Distributed Computation Offloading

The key idea of the self-learning based distributed computation offloading (DISCO) algorithm is making full use of the FIP of $\mathcal{G}$. In view of this, a finite number of offloading decision updating iterations can achieve a plateau status. Moreover, to make the best response decision, a vehicle needs to know the computation offloading decisions of other vehicles according to Lemma 1 and formula (20). To this end, we utilize a message exchange protocol [27]–[30], where vehicles that have the best response decisions compete for the decision updating opportunity in a distributed manner and only one decision is made at a time. More specifically, the best response updating set of vehicle n is first calculated out according to Lemma 1 and formula (20) as

$$\begin{aligned} \Upsilon_n &\triangleq \{\chi_n^* : U_n(\chi_n^*, \boldsymbol{\chi}_{-n}) < U_n(\chi_n, \boldsymbol{\chi}_{-n})\} \\ &= \begin{cases} \{1\}, & \text{if } \chi_n = 0 \text{ and } \sum_{i\in\mathcal{N}\setminus\{n\}} \chi_i < \Gamma_n - 1, \\ \{0\}, & \text{if } \chi_n = 1 \text{ and } \sum_{i\in\mathcal{N}\setminus\{n\}} \chi_i \geq \Gamma_n - 1, \\ \varnothing, & \text{otherwise.} \end{cases} \end{aligned} \tag{51}$$

Then, according to the set of best response updating Υ_n, vehicle n decides weather to compete for the decision updating opportunity. That is,

- if $\Upsilon_n \neq \varnothing$, vehicle n will compete for the decision updating opportunity;
- otherwise, vehicle n will not compete. Instead, it will keep the current decision unchanged.

To address the potential collisions if multiple vehicles compete for the decision updating opportunity simultaneously, we adopt a random backoff-based mechanism [40]. Specifically, we set the time duration of decision updating competition as τ. Each competing vehicle initializes a timer with a backoff time duration σ_n that obeys a uniform distribution on the interval $[0, \tau]$ and countdowns the timer. If a competing vehicle has not received any request-to-updating (RTU) message from other vehicles yet when the timer expires,

the vehicle will update its decision as defined in formula (51) and broadcast an RTU message to all other vehicles. If other competing vehicles receive the RTU message, they will give up the updating opportunity and keep their current decisions. It is worth noting that each RTU message includes the ID of the vehicle and its decision. The broadcasting of RTU message can be easily achieved by the control channel of LTE-V communication protocol. And each vehicle keeps a record memory $\mathcal{M}$ to record decisions of other vehicles based on the received RTU message.

Algorithm 1 Self-Learning Based Distributed Computation Offloading (DISCO) Algorithm

1: **Initialization**
2: Initialize the decision of each vehicle as $\chi_n = 0$
3: Initialize record memory of each vehicle as $\mathcal{M}_n = \{0, \ldots\}$
4: Calculate the initial value of joint cost U_n of each vehicle
5: **End Initialization**
6: **Begin**
7: **for** iteration $k = 1, 2, 3, \ldots$:
8: **for** each vehicle n in parallel:
9: **do**:
10: Obtain decision profile $\boldsymbol{\chi}$ based on $\mathcal{M}_n$
11: Calculate Υ_n according to formula (51)
12: **if** $\Upsilon_n \neq \varnothing$ **then**
13: Compete for the decision updating opportunity
14: **if** win the competition opportunity **then**
15: Choose the decision in Υ_n
16: Broadcast the RTU message
17: **else**
18: Keep χ_n unchanged
19: **end if**
20: **else**
21: Keep χ_n unchanged at the next iteration
22: **end if**
23: **until** no RTU message is broadcasted
24: **end for**
25: **end for**
26: **for** each vehicle n in parallel:
27: Execute the computation offloading decision χ_n obtained at the last iteration
28: **end for**
29: **End**

According to the FIP, after a finite number of iterations, the formulated game $\mathcal{G}$ would converge to a Nash Equilibrium. At the Nash Equilibrium point, no vehicle would change its decision thus no RTU message would be broadcasted. In view of this, we judge that the iterations can terminate if no RTU message is broadcasted. Algorithm 1 presents the detailed DISCO algorithm consisting of *Self-Learning Stage* and *Executing Stage*. At the *Self-Learning Stage*, as shown in Lines 1-25 of Algorithm 1, each vehicle learns from its individual information and the decision profile of other vehicles to make the best response decision through broadcasting RTU message. At the *Executing Stage*, as shown in Lines 26-28 of Algorithm 1, each vehicle executes its best response decision learned at the *Self-Learning Stage*.

It is obvious that the *Self-Learning Stage* impacts the efficiency of the proposed DISCO algorithm the most. For each iteration, only basic arithmetical calculations will be executed by N vehicle as shown in in Lines 10-22 of Algorithm 1. Accordingly, the computational complexity for each iteration is $\mathcal{O}(N)$. Since the iteration will terminate after a finite number of iterations according to the FIP of game $\mathcal{G}$, let K denote the number of iterations for DISCO to converge. Accordingly, $\mathcal{O}(KN)$ is the total computational complexity of DISCO. Since the computational time is very short, generally several microseconds, this part of time can be ignored. Accordingly, the time length of each iteration during the *Self-Learning Stage* is mainly depending on the time duration of decision updating competition and the RTU message broadcasting time. For the time duration of decision updating competition, since τ can be very short, generally several milliseconds, we set $\tau = 10$ ms in our paper. For the RTU message broadcasting time, since it depends on the size of RTU message and such RTU message that only containing the user's ID and decision is very small, generally several milliseconds [41]. Accordingly, the proposed DISCO algorithm has a fast convergence, which will be verified in the following.

It is noting that vehicles may enter and leave the coverage of the RSU during self-learning and task offloading, leading to a failed result reception from RSU. To address this issue, we can first adopt a duration prediction method to evaluate the link duration between vehicles and RSU based on the current position and speed of vehicles, and the position of RSU. Then, a threshold duration is set, guaranteeing the task processing result can be returned before the vehicle leaves the coverage of the RSU. Only the vehicles whose predicted link duration is longer than the threshold duration can participate in the gaming and task offloading process. Another way is utilizing the cooperation between adjacent RSUs. The remaining task data can be offloaded to the next RSU if the vehicle leaves the coverage of the current RSU during task offloading, or can be migrated from the current RSU to the next RSU if the vehicle leaves the coverage of the current RSU during task processing. For simplicity, in this paper, we assume that all vehicles with offloading tasks are still in the coverage range of the current RSU during the period from the beginning of self-learning process to the time when results are received.

VI. Simulation Results and Discussions

In this section, we conduct simulations to validate the performance of the proposed DISCO algorithm. We first describe the simulation setup and then discuss the simulation results.

A. Simulation Setup

We consider a two-way two-lane scenario. The length and width of each lane are 1000 m and 4 m, respectively. And one RSU is deployed in the middle of roadside, with coordinate (0, 0). The trajectory of vehicles is randomly chosen from GAIA Open Dataset of DiDi Express [42]. The coverage radius of RSU and vehicles are set to 500 m and 250 m,

TABLE II
PARAMETERS SETTING ABOUT VEHICLES AND RSU

Description	Value
Bandwidth per channel (B)	0.5 MHz
Processing density of data (C_n)	100
Data size of T_n (D_n)	$1 \sim 10$ Mbit
Processing capability of each core (f^e)	1×10^9
Processing capability of vehicle (f_n^l)	1.4×10^8
Switched capacitance coefficient (κ_n, κ_e)	10^{-24}, 10^{-29}
Number of V2I channels (L)	$10 \sim 50$
Number of processing cores of RSU (M)	64
Number of vehicles (N)	$10 \sim 60$
Transmission power of vehicles (P_n)	30 dBm
Coverage radius of vehicles	250 m
Coverage radius of RSU	500 m
White Gaussian noise power (ω_0)	-100 dBm
Path loss exponent (ϑ)	4
Energy consumption cost coefficient (ϱ)	2.44×10^{-4}
V2I cost coefficient (ξ)	1.16×10^{-1}
Cost for RSU processing data (γ)	3×10^{-9}
Weighting parameter of latency (λ_1)	$0 \sim 1$
Weighting parameter of cost (λ_2)	$0 \sim 1$
Maximal backoff time duration (τ)	10 ms

respectively. The data size is randomly distributed between 1 and 10 Mbit. The detailed parameters setting about vehicles and RSU is shown in Table II. We use a GPU-based server with 4 NVIDIA GTX2080 Ti GPUs, where the CPU is Intel Xeno(R) E5-2690v4 with 64 G memory. Software environment we utilize is Python 3.7 on Ubuntu16.04.6 LTS.

B. Simulation Results

We consider the following schemes as benchmarks to evaluate our proposed DISCO:

- Offload-Comp-Only (OCO), where all vehicles offload their computation tasks to RSU to be processed;
- Local-Comp-Only (LCO), where all vehicles compute their computation tasks locally.
- Random-Offload (RO), where each vehicle choose decision 1 or 0 randomly.
- JROPSO, which is a centralized computational offloading decision algorithm based on particle swarm optimization (PSO) [43].

1) Effectiveness: We first evaluate the timeliness and effectiveness of the proposed DISCO algorithm. In this set of simulations, we set $N = 40$, $L = 30$, $\lambda_1 = \lambda_2 = 0.5$, D_n obeys an uniform distribution on interval [1, 10] Mbit. Fig. 2 shows the changes in the system-wide joint cost (i.e., $\sum_{n\in\mathcal{N}} U_n(\chi_n, \boldsymbol{\chi}_{-n})$) obtained by iterations under different schemes. The figure indicates that the system-wide joint cost of DISCO tends to be optimal and stable in about 50 iterations. As analyzed in Section V, since the time length of each iteration is very short (generally in the unit of millisecond), the convergence time of the proposed DISCO is also very short. Accordingly, the proposed DISCO algorithm has

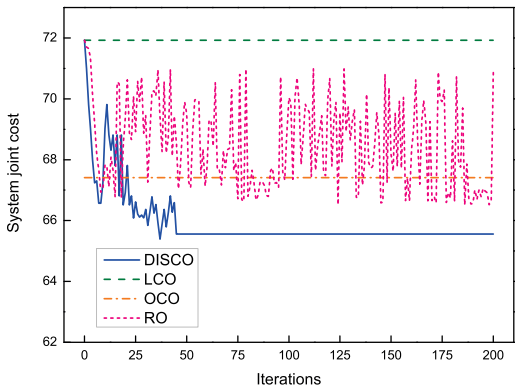


Fig. 2. System-wide joint cost achieved during iterations.

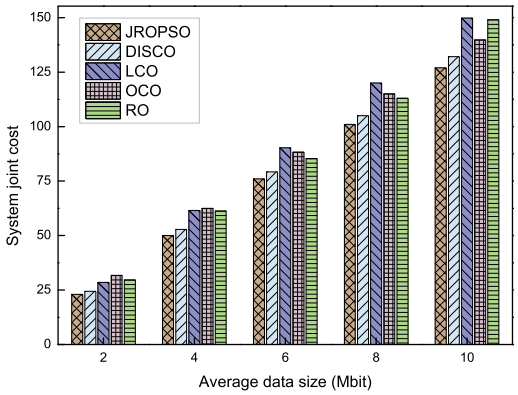


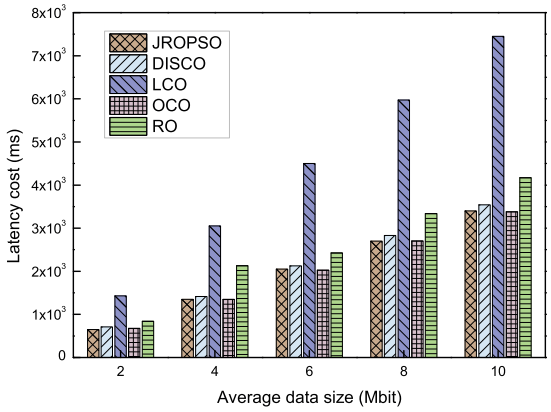
Fig. 3. System joint cost under different average data size.

a fast convergence. Moreover, the system joint cost of DISCO is lower than both LCO and OCO schemes. This is because the vehicles in our proposed DISCO continue to learn the best response decision to minimize their joint costs. However, a high joint cost will be produced if the task with big data size is computed locally by the LCO scheme or if the task with small data size is offloaded to RSU to be processed by the OCO scheme. Besides, the system joint cost of the RO scheme fluctuates dramatically over iterations and higher than that of the proposed DISCO algorithm.

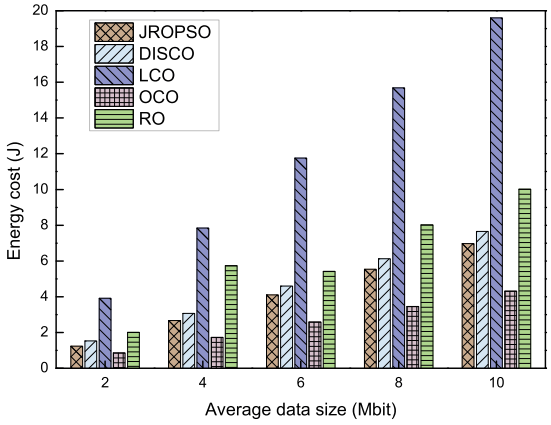
2) System Joint Cost: Fig. 3 shows the relationship between system joint cost and average data size when we set $N = 40$, $L = 30$, and $\lambda_1 = \lambda_2 = 0.5$. It can be seen that the system joint costs of all schemes increase with the increasing of average data size. Because a larger data size requires more communication (if offloaded) and computing costs. The proposed DISCO outperforms LCO, OCO and RO, and can reduce up to 14%, 23% and 18% system joint cost over the three schemes, respectively. Compared with the centralized JROPSO, the performance loss of DISCO is less than 7%. Moreover, the performance of OCO is worse than LCO when average data size is small and becoming better than LCO when average data size is getting bigger. This is because when data size is small, computing time cost of local computing will be smaller compared with the cost of RSU computing. And when data size is big, severe computing time will be caused by local

TABLE III
THE NUMBER OF VEHICLES CHOOSING DECISION 0 OR 1 WITH DIFFERENT AVERAGE DATA SIZE OF DISCO

Average data size (Mbit)	2	4	6	8	10
Number of vehicles choosing decision 0	30	24	11	7	4
Number of vehicles choosing decision 1	10	16	29	33	36



(a) Latency cost



(b) Energy cost

Fig. 4. Latency cost and energy cost under different average data size.

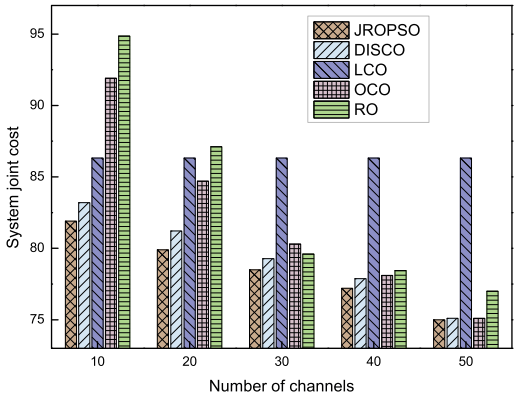


Fig. 5. System joint cost under different number of channels.

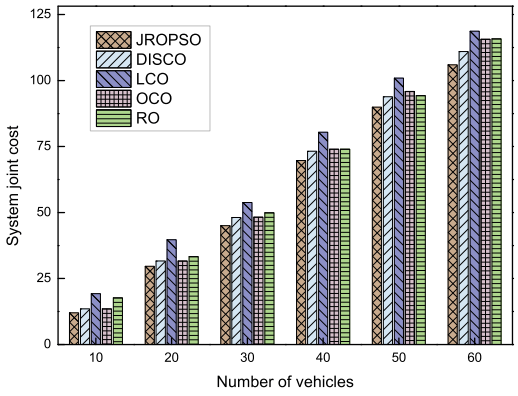


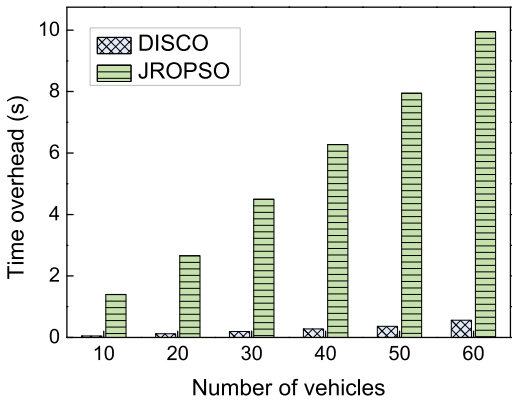
Fig. 6. System joint cost under different number of vehicles.

computing thus results in a higher joint cost compared with RSU computing. To verify this point, we further show how the number of vehicles choosing decision 0 and 1 changes under different average data size of our proposed DISCO. As shown in Table III, the number of vehicles choosing decision 1 increases while the number of vehicles choosing decision 0 decreases as the average data size increases.

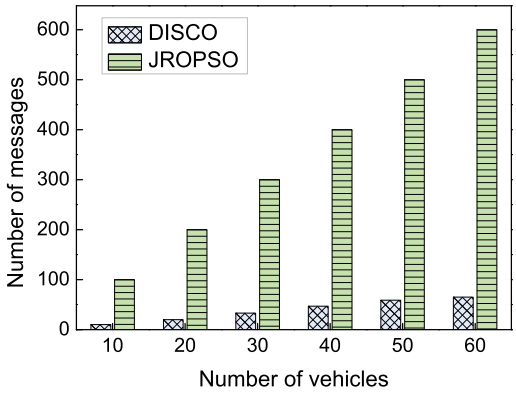
Since the system joint cost is a weighted system cost, it is hard to understand the cost of each component. To provide a more straightforward understanding, we present in Fig. 4 the real value of latency and energy cost of Fig. 3. For the latency cost, as shown in Fig. 4(a), the LCO has the worst performance since the limited local computation capacity causing long computing time. For the latency cost, as shown in Fig. 4(b), the LCO also has the worst performance since long latency causing more energy corruption.

Fig. 5 shows how communication resource (i.e., the number of channels L) impacts the system joint cost when we set $N = 40$, $\lambda_1 = \lambda_2 = 0.5$, and D_n obeys an uniform distribution on interval [1, 10] Mbit. The system joint costs of all schemes except LCO decrease with the increasing L. Because more communication resources will be obtained for vehicles choosing decision 1 if L increases, thus transmission time cost is reduced. The proposed DISCO outperforms LCO, OCO, and RO, which can reduce up to 13%, 10%, and 12% system joint cost over the three schemes, respectively, with less than 3% performance loss compared with JROPSO.

Fig. 6 shows how the number of vehicles impacts the system joint cost when we set $L = 30$, $\lambda_1 = \lambda_2 = 0.5$, and D_n obeys an uniform distribution on interval [1, 10] Mbit. It shows the system joint costs of all schemes increase with the increasing number of vehicles. The proposed DISCO outperforms LCO, OCO, and RO, which can reduce up to 29%, 5%, and 23% system joint cost over the three schemes, respectively, with less than 12% performance loss compared



(a) Time overhead



(b) Message overhead

Fig. 7. Time overhead and message overhead.

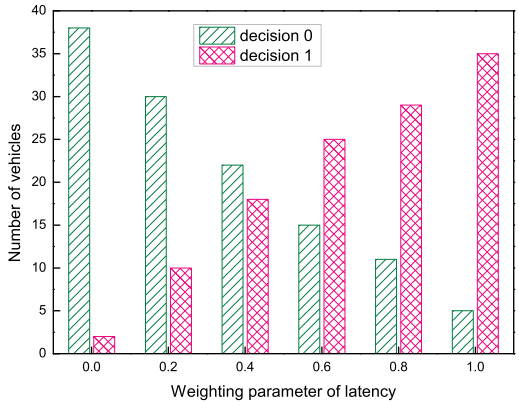


Fig. 8. Offloading choices under different λ_1 and λ_2.

with JROPSO. Moreover, the performance gap between DISCO and OCO is increasing with the number of vehicles. Because the average communication resource each vehicle can obtain is getting smaller if the number of vehicles increases for the OCO scheme, thus severe transmission time cost will be produced. The vehicles in our proposed DISCO can choose decision 0 or 1 intelligently towards a minimum joint cost.

3) Overhead: We evaluate the overhead performance between DISCO and JROPSO on two aspects, i.e., the algorithm execution time overhead (hereinafter referred to as *time overhead* for short) and controlling and signaling overhead (i.e., the number of messages exchanged among vehicles and between vehicles and RSU, hereinafter referred to as *message overhead* for short).

Fig. 7(a) shows the time overhead comparison when we set $L = 30$, $\lambda_1 = \lambda_2 = 0.5$, and D_n obeys an uniform distribution on interval [1, 10] Mbit. It demonstrates that the time overhead of DISCO is much lower than that of JROPSO. The order-of-magnitude of DISCO's time overhead is in milliseconds while the order-of-magnitude of JROPSO's time overhead is in seconds. Accordingly, DISCO can achieve an order-of-magnitude improvement on time overhead over JROPSO. The reasons are twofold. The first one is that massive information from all vehicles needs to be collected, processed and calculated to obtain a global optimal offloading scheme by continuous iterations for JROPSO. For DISCO, each vehicle only needs the decision profile of other vehicles and calculates its best response decision in a distributed and parallel way, which reduces the time overhead. Most importantly, the JROPSO algorithm itself, as a centralized algorithm, has a higher computational complexity as many centralized heuristic algorithm do. As analyzed in Section V, only basic arithmetical calculations will be executed by N vehicles for each iteration, the computational complexity is very low. However, for JROPSO algorithm, massive calculations will be executed in each iteration since the number of particles and the scale of the problem jointly determine the computational complexity. And more iterations are needed for JROPSO algorithm to converge, leading to a much longer convergence time.

Fig. 7(b) shows message overhead comparison when we set $L = 30$, $\lambda_1 = \lambda_2 = 0.5$, and D_n obeys an uniform distribution on interval [1, 10] Mbit. It can be seen that the message overhead of DISCO is much lower than that of JROPSO. Specifically, DISCO can reduce message overhead by at least 88% over JROPSO under different numbers of vehicles. The reason is that for DISCO, message overhead is produced only when a vehicle successfully wins the decision updating opportunity and updates its offloading decision. While for JROPSO, all information of each vehicle is needed to send to RSU, including the task data size, the processing density of data, the local processing density, the transmission power, the position, and many other parameters. There is another main concern for JROPSO that some vehicles may be unwilling to send their information due to privacy concerns and hence are unwilling to participate in the centralized optimal offloading algorithm. And our proposed DISCO has the advantage of not having to consider the privacy issue. Because each vehicle can make the best response decision locally without exposing its local parameters.

4) Weighting Parameters λ_1 and λ_2: Fig. 8 shows how different λ_1 and λ_2 impact vehicles' offloading decisions. In this set of simulation, we set $N = 40$, $L = 30$, $\lambda_1 \in [0, 1]$ $\lambda_2 \in [0, 1]$, and $\lambda_1 + \lambda_2 = 1$. More vehicles tend to choose local computing when the weighting parameter of latency λ_1 is small. In this case, we emphasize more on a lower cost. Offloading tasks to the RSU would produce more costs by using communication and computation resources.

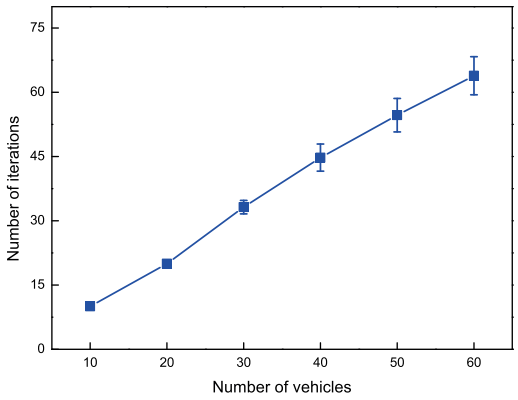


Fig. 9. Number of iterations.

Since our proposed DISCO tries to minimize the joint cost, more vehicles gradually learn that they should choose a decision 0. When the weighting of latency λ_1 is becoming bigger (i.e., IoV applications seek a lower latency rather than cost), more vehicles tend to offload their tasks to the RSU. In this case, the powerful computing capacity of RSU can reduce the computing time if a decision 1 is chosen. More vehicles would gradually choose to offload their tasks to RSU for minimizing the joint cost by our proposed DISCO.

5) Convergence Under Different Number of Vehicles: Fig. 9 shows the average number of iterations for DISCO to converge to a Nash Equilibrium. As shown, the average number of iterations increases linearly with the number of vehicles, which indicates that our proposed DISCO scales well with the number of vehicles. This is another advantage of DISCO over the centralized offloading algorithms whose computational complexities usually increase exponentially with the number of vehicles.

VII. Conclusion

In this paper, we have investigated the computation offloading problem and proposed a self-learning based distributed computation offloading scheme for IoV. Specifically, we first established an offloading framework with communication and computation for IoV. Then we formulated a distributed computation offloading game, where each vehicle as a player makes an offloading decision to minimize its own joint cost. We proved that our formulated distributed computation offloading game admits a Nash Equilibrium. To get to the Nash Equilibrium of the formulated game, we designed a self-learning based distributed computation offloading (DISCO) algorithm. Since no control center is required, DISCO is a very practical algorithm. Finally, we used real-world vehicular traces to implement extensive simulations, which verified the performance of DISCO. Our proposed DISCO scheme can achieve at least an order-of-magnitude improvement on time overhead and 88% performance gain on message overhead, only at up to 12% performance loss on system joint cost, compared with the centralized scheme.

In the future, we will consider continuous tasks and utilize the dynamic game to analyze the offloading decision-making problem for a more practical Internet of vehicle scenario. Also, we should consider the benefits obtained by service providers thus contract-based method should be combined with a game theory-based method. The mobility of vehicles and the handover between different RSUs should be also considered.

References

[1] 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.

[2] H. Zhou, W. Xu, J. Chen, and W. Wang, "Evolutionary V2X technologies toward the Internet of Vehicles: Challenges and opportunities," *Proc. IEEE*, vol. 108, no. 2, pp. 308–323, Feb. 2020.

[3] S. Sharma and B. Kaushik, "A survey on Internet of Vehicles: Applications, security issues & solutions," *Veh. Commun.*, vol. 20, Dec. 2019, Art. no. 100182.

[4] N. Lu, N. Cheng, N. Zhang, X. Shen, and J. W. Mark, "Connected vehicles: Solutions and challenges," *IEEE Internet Things J.*, vol. 1, no. 4, pp. 289–299, Aug. 2014.

[5] L. Lin, X. Liao, H. Jin, and P. Li, "Computation offloading toward edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1584–1607, Aug. 2019.

[6] J. E. Siegel, D. C. Erb, and S. E. Sarma, "A survey of the connected vehicle Landscape—Architectures, enabling technologies, applications, and development areas," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 8, pp. 2391–2406, Aug. 2018.

[7] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[8] 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.

[9] 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.

[10] S. Wang *et al.*, "When edge meets learning: Adaptive control for resource-constrained distributed machine learning," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 63–71.

[11] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proc. IEEE*, vol. 107, no. 8, pp. 1738–1762, Aug. 2019.

[12] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the Internet of Vehicles," *Proc. IEEE*, vol. 108, no. 2, pp. 246–261, Feb. 2020.

[13] K. Zhang, Y. Mao, S. Leng, A. Vinel, and Y. Zhang, "Delay constrained offloading for mobile edge computing in cloud-enabled vehicular networks," in *Proc. 8th Int. Workshop Resilient Netw. Design Modeling (RNDM)*, Sep. 2016, pp. 288–294.

[14] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1079–1092, Feb. 2019.

[15] C. Wang, C. Liang, F. R. Yu, Q. Chen, and L. Tang, "Computation offloading and resource allocation in wireless cellular networks with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 4924–4938, Aug. 2017.

[16] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[17] K. Zhang, S. Leng, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1987–1997, Apr. 2019.

[18] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

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

[20] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[21] D. Monderer and L. S. Shapley, "Potential games," *Games Econ. Behav.*, vol. 14, no. 1, pp. 124–143, 1996.

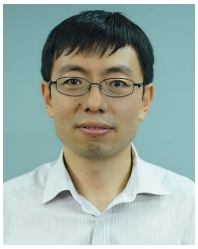
[22] J. Li, G. Kendall, and R. John, "Computing Nash equilibria and evolutionarily stable states of evolutionary games," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 460–469, Jun. 2016.
[23] X. Huang, K. Xu, C. Lai, Q. Chen, and J. Zhang, "Energy-efficient offloading decision-making for mobile edge computing in vehicular networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, p. 35, Dec. 2020.
[24] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.
[25] J. Zhang, J. Du, Y. Shen, and J. Wang, "Dynamic computation offloading with energy harvesting devices: A hybrid-decision-based deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9303–9317, Oct. 2020.
[26] 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.
[27] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.
[28] 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.
[29] H. Cao and J. Cai, "Distributed multiuser computation offloading for cloudlet-based mobile cloud computing: A game-theoretic machine learning approach," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 752–764, Jan. 2018.
[30] M.-A. Messous, S.-M. Senouci, H. Sedjelmaci, and S. Cherkaoui, "A game theory based efficient computation offloading in an UAV network," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4964–4974, May 2019.
[31] T. Q. Dinh, Q. D. La, T. Q. Quek, and H. Shin, "Learning for computation offloading in mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 12, pp. 6353–6367, Aug. 2018.
[32] Y. Sun, S. Zhou, and Z. Niu, "Distributed task replication for vehicular edge computing: Performance analysis and learning-based algorithm," 2020, *arXiv:2002.08833*. [Online]. Available: http://arxiv.org/abs/2002.08833
[33] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2457–2467, Nov. 2017.
[34] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.
[35] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.
[36] H. Guo and J. Liu, "UAV-enhanced intelligent offloading for Internet of Things at the edge," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2737–2746, Apr. 2020.
[37] J. Wallenius, J. S. Dyer, P. C. Fishburn, R. E. Steuer, S. Zionts, and K. Deb, "Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead," *Manage. Sci.*, vol. 54, no. 7, pp. 1336–1349, Jul. 2008.
[38] Y. Kim, J. Kwak, and S. Chong, "Dual-side optimization for cost-delay tradeoff in mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 2, pp. 1765–1781, Feb. 2018.
[39] M. Felegyhazi and J.-P. Hubaux, "Game theory in wireless networks: A tutorial," EPFL, Lausanne, Switzerland, Tech. Rep. LCA-REPORT-2006-002, 2006.
[40] Q. Zhao, L. Tong, A. Swami, and Y. Chen, "Decentralized cognitive MAC for opportunistic spectrum access in ad hoc networks: A POMDP framework," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 3, pp. 589–600, Apr. 2007.
[41] Y. Xu, J. Wang, Q. Wu, A. Anpalagan, and Y.-D. Yao, "Opportunistic spectrum access in unknown dynamic environment: A game-theoretic stochastic learning solution," *IEEE Trans. Wireless Commun.*, vol. 11, no. 4, pp. 1380–1391, Apr. 2012.
[42] Didi. *Urban Traffic Time Index and Trajectory Data (New)*. Accessed: Oct. 22, 2019. [Online]. Available: https://gaia.didichuxing.com
[43] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: A particle swarm optimization approach," *Appl. Sci.*, vol. 10, no. 1, p. 203, Dec. 2019.

Quyuan Luo received the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2020. From 2019 to 2020, he was a Visiting Scholar with the Department of Computer Science, Wayne State University, USA. He is currently an Assistant Professor with the School of Information Science and Technology, Southwest Jiaotong University. His current research interests include intelligent transportation systems, content distribution, edge computing, and resource allocation in vehicular networks.

Changle Li (Senior Member, IEEE) received the Ph.D. degree in communication and information system from Xidian University, Xi'an, China, in 2005. He conducted his postdoctoral research in Canada and in the National Institute of information and Communications Technology, Japan. He was a Visiting Scholar with the University of Technology Sydney. He is currently a Professor with the State Key Laboratory of Integrated Services Networks, Xidian University. His research interests include intelligent transportation systems, vehicular networks, mobile ad hoc networks, and wireless sensor networks.

Tom H. Luan (Senior Member, IEEE) received the B.E. degree from Xi'an Jiaotong University, Xi'an, China, in 2004, the M.Phil. degree from the Hong Kong University of Science and Technology, Hong Kong, in 2007, and the Ph.D. degree from the University of Waterloo, Waterloo, ON, Canada, in 2012. From 2013 to 2017, he was a Lecturer with the School of Information Technology, Deakin University, Burwood, VIC, Australia. He is currently a Professor with the School of Cyber Engineering, Xidian University. His current research interests include content distribution in vehicular networks, mobile cloud computing, and fog computing.

Weisong Shi (Fellow, IEEE) received the B.S. degree in computer engineering from Xidian University, Xi'an, China, in 1995, and the Ph.D. degree in computer engineering from the Chinese Academy of Sciences, Beijing, China, in 2000. He is currently a Charles H. Gershenson Distinguished Faculty Fellow and a Professor of Computer Science with Wayne State University, Detroit, MI, USA. His current research interests include edge computing, computer systems, energy-efficiency, and wireless health. He was a recipient of the National Outstanding Ph.D. Dissertation Award of China and the NSF CAREER Award. He is an ACM Distinguished Scientist.



Weigang Wu (Member, IEEE) received the B.Sc. and M.Sc. degrees from Xi'an Jiaotong University, China, in 1998 and 2003, respectively, and the Ph.D. degree in computer science from The Hong Kong Polytechnic University in 2007. He is currently a Full Professor with the School of Data and Computer Science, Sun Yat-sen University, China. He has published more than 60 papers in major conferences and journals. His research interests include distributed systems and wireless networks, especially cloud computing platforms, and ad hoc networks. He has served as a member for the Editorial Board of two international journals, *Frontiers of Computer Science* and *Ad Hoc & Sensor Wireless Networks*.