Fully Distributed Task Offloading in Vehicular Edge Computing

Qianpiao Ma, Hongli Xu, *Member, IEEE*, Haibo Wang, *Member, IEEE*, Yang Xu, *Member, IEEE*, Qingmin Jia, and Chunming Qiao, *Fellow, IEEE*

***Abstract*—In vehicular edge computing (VEC), the deployment of road side units (RSUs) along roads enables vehicles to offload computation-intensive tasks for efficient data processing. However, VEC poses unique challenges, including resource constraints on vehicles and RSUs, high vehicle mobility, and the large-scale nature of the infrastructure. Existing solutions, whether centralized or distributed, often suffer from longer decision-making times or task response times, making them unsuitable for vehicular scenarios. To address these challenges, this paper proposes a Fully Distributed Task Offloading (FDTO) decision-making scheme, which enables vehicles to iteratively adjust their offloading decisions based on resource utilization information obtained from neighboring RSUs. FDTO employs two different algorithms for decision adjustments: a greedy-based algorithm and a convex optimization-based algorithm. Theoretical analysis proves the convergence of the proposed algorithms to a global optimum through iterations. To evaluate the performance of FDTO, extensive simulations are conducted and the results demonstrate that the proposed algorithms offer near-optimal performance with a short decision-making time, reducing the average task response time by 50%-65% compared to existing algorithms.**

***Index Terms*—Vehicular edge computing (VEC), distributed task offloading, delay optimal, decision making.**

I. Introduction

THE increasing popularity of the Internet of Things (IoT) and Artificial Intelligence (AI) is propelling the advancement of vehicular networks [1]. These networks are designed to collect data from various equipment on mobile vehicles, including cameras and radars, to enable the development of intelligent applications in future transportation systems. These applications, such as path planning [2] and vehicle recognition [3], are crucial for alleviating traffic congestion, enhancing road safety, and improving the overall quality of service (QoS). It is imperative that the results obtained from these applications are not only efficient but also delivered promptly to ensure optimal performance across a wide range of vehicular network.

In traditional vehicular networks, a significant volume of application data is transmitted to a remote cloud platform via the core network, thereby imposing stringent demands on transmission bandwidth and resulting in delays in responding to application requests [4]. This situation is further exacerbated when the core network experiences congestion or heavy traffic, rendering response times unpredictable, especially in scenarios with high traffic dynamics. Consequently, none of these cases meet the stringent requirement for low response time.

Mobile Edge Computing (MEC) [5] has been proposed to address the escalating demands for computation and alleviate the burden on core networks. In this approach, the processing capacity is shifted from the cloud to the edge, enabling application tasks to be executed at network edges [6], [7], [8]. This advancement has paved the way for the emergence of Vehicular Edge Computing (VEC) [9], which involves offloading computationally intensive tasks from vehicles to Road Side Units (RSUs) equipped with communication, computation, and storage resources. However, to design an efficient task offloading decision-making mechanism for VEC, we should take the following unique characteristics of vehicular networks into considerations. 1) Resource constraints on vehicles. For example, compared with large-scale data centers, the computation resources on RSUs are very limited. 2) High dynamics. Due to high mobility of vehicles and dynamic task arrival, a long decision making time may make the offloading decision infeasible. That is, only real-time offloading decisions make sense. 3) Large scale. It is a great challenge to make suitable offloading decisions in a large-scale vehicular network with enormous amount of vehicles and RSUs.

Efforts have been dedicated to achieving efficient task offloading in VEC, which can be broadly classified into two types. The first type is centralized task offloading [10], [11], [12], [13], where an entity within the VEC infrastructure, typically the remote cloud platform or an RSU, acts as the controller and gathers all the necessary information to make global task offloading decisions for all vehicle tasks. For instance, [10] formulates task offloading as a mixed integer

Manuscript received 28 May 2023; revised 5 August 2023, 15 September 2023, and 28 October 2023; accepted 6 November 2023. Date of publication 9 November 2023; date of current version 22 April 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 92267301 and in part by Jiangning Baijia Lake Plan Program under Grant 74072203-3. The review of this article was coordinated by Dr. Chau Yuen. *(Corresponding authors: Qianpiao Ma; Qingmin Jia.)*

Qianpiao Ma and Qingmin Jia are with Purple Mountain Laboratories, Nanjing, Jiangsu 211111, China (e-mail: maqianpiao@pmlabs.com.cn; jiaqingmin@pmlabs.com.cn).

Hongli Xu and Yang Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China (e-mail: xuhongli@ustc.edu.cn; xuyangcs@ustc.edu.cn).

Haibo Wang is with the Department of Computer Science, University of Kentucky, Lexington, KY 40506 USA (e-mail: wanghaibo@uky.edu).

Chunming Qiao is with the Department of Computer Science and Engineering, State University of New York, Buffalo, NY 14260 USA (e-mail: qiao@computer.org).

Digital Object Identifier 10.1109/TVT.2023.3331344

TABLE I
COMPARISON OF DIFFERENT OFFLOADING DECISION SOLUTIONS

Solution	Task Response Time	Decision Making Time	Handle Dynamics	Scalability
Centralized [10]–[13]	Short	Long	Poor	Poor
Game-based Distributed [14]–[17]	Short	Long	Poor	Poor
Selfish Distributed [18] [19]	Long	Short	Good	Good
Iterative Distributed (**Ours**)	Short	Short	Good	Good

non-linear (MINLP) program and solves it to obtain the offloading strategy for vehicles through centralized decision-making. Due to the high complexity in solving MINLP program, [11], [12], [13] provide learning-based methods for decision making in VEC. However, these centralized solutions has several limitations. Firstly, it is susceptible to single-point failures, where the controller device malfunction can render the entire system non-functional and require a complete reset. Secondly, the limited computation capacity of the centralized controller becomes a bottleneck as the VEC scale expands, resulting in slower decision-making.

The second type of solutions in VEC is distributed task offloading, where each vehicle autonomously makes offloading decisions. Some existing distributed decision-making solutions are based on game theory [14], [15], [16], [17]. For example, [17] formulates task offloading as a distributed decision making game, where each vehicle, as a player, makes its best response decision by a self-learning based algorithm. Although these algorithms overcome the single-point failure problem of the centralized solutions, it may not be practical for large scale VEC. The reason being, they require vehicles to take turns executing algorithms in a round-robin fashion until a Nash equilibrium is reached, incurring a time complexity that scales with the number of vehicles, making it impractical for time-sensitive applications. Another kind of distributed solutions involve each vehicle making selfish offloading decisions based solely on its local information [18], [19], enabling real-time offloading decisions. However, these solutions may suffer from suboptimal performance due to the lack of global information. For example, an RSU may face severe congestion when too many vehicles choose to offload tasks to it. Consequently, achieving an optimal offloading strategy or even a satisfactory performance level remains challenging.

In this paper, we propose a fully distributed iterative scheme for task offloading in VEC, aiming to achieve short task response time with fast offloading decisions. Specifically, each vehicle needs only collect resource information from the neighbour RSUs [17], [20]. Through an iterative interaction between RSUs and vehicles, each vehicle can independently make a offloading decision with low complexity, while achieving a theoretic global optimum. Compared to the centralized and game-based distributed solutions, our solution significantly reduces decision making time and remains unaffected by the scale of VEC, ensuring real-time offloading decisions for vehicles and enabling better scalability. Compared to the selfish distributed solutions, our iterative scheme allows each vehicle to consider the decisions of other vehicles when offloading tasks, enabling better global performance. A summarized comparison of offloading solutions is provided in Table I. The main contributions of this paper can be summarized as follows:

1) We formulate the delay-optimal task offloading problem in VEC and propose a fully distributed task offloading (FDTO) scheme, where vehicles make their offloading decisions only based on its neighboring information rather than the whole network, significantly improving the scalability for dealing with system dynamics.
2) We propose a distributed iterative algorithm for task offloading, which consists of two algorithms for the RSU side and the vehicle side, respectively. In particular, we design two vehicle side algorithms to adjust their offloading decisions. One is the greedy based algorithm FDTO-VG, with very simple computation, and the other is called FDTO-VX based on convex optimization. We also theoretically prove that the proposed algorithms will converge to the global optimum.
3) We conduct large-scale simulations for numerical evaluation. The simulation results show that our proposed algorithms can reduce the average task response time by about 50%–65% compared with the existing algorithms while keeping a short decision making time in a large scale MEC.

II. System Model

A. Network Model

Consider a VEC-enabled system comprising a collection of n vehicles, denoted as $V = \{v_1, v_2, \ldots, v_n\}$, where $n = |V|$, and a set of m Road Side Units (RSUs), denoted as $S = \{s_1, s_2, \ldots, s_m\}$, where $m = |S|$. These vehicles generate tasks for various applications, such as path planning and vehicle recognition. The task arrivals on vehicle v_i follow a Poisson process with an expected arrival rate of ϕ_i [5]. The total task arrival rate in the system is represented as $\Phi = \sum_{v_i \in V} \phi_i$. Tasks can vary in terms of input data size, output data size, and the required number of CPU cycles for execution. The expected values of these parameters on any vehicle v_i are denoted as ζ_i, θ_i, and δ_i, respectively. Each RSU $s_j \in S$ has a maximum computation capacity, denoted as μ_j, which is measured in terms of the number of CPU cycles per unit time. Additionally, each RSU s_j is assigned an energy consumption budget per unit time, represented as $E_j^{\max}$. Vehicles offload their tasks to RSUs for efficient processing. $p_{i,j}$ denotes the probability that vehicle v_i offloads its tasks to RSU s_j. Some important notations of this paper are listed in Table II.

TABLE II
NOTATION SUMMARY

Symbol	Semantics
V	A set of vehicles $\{v_1, v_2, ..., v_n\}$
S	A set of RSUs $\{s_1, s_2, ..., s_m\}$
E_j	Energy consumption per unit time on s_j
E_j^{max}	Energy budget per unit time on s_j
V_j	Set of vehicles which can communicate with s_j
S_i	Set of RSUs that v_i communicates with
Φ	Total task arrival rate
ϕ_i	Task arrival rate on v_i
δ_i	Expected number of required CPU cycles of tasks from v_i
ζ_i	Expected input data size of tasks from v_i
θ_i	Expected downlink data size of tasks back to v_i
$r_{i,j}^v$	Uplink transmission rate from v_i to s_j
$r_{i,j}^d$	Downlink transmission rate from s_j to v_i
$\xi_{i,j}$	Required computing capacity on s_j from v_i
$\eta_{i,j}$	Downlink traffic from s_j to v_i
λ_j	Required computing capacity on s_j
μ_j	Computing capacity of s_j
κ_j	Energy consumption for executing one CPU cycle on s_j
τ	The length of one time slot
$p_{i,j}$	The offloading probability

B. Transmission Model

1) Limited Range Transmission: Let P_i^u represent the transmission power of vehicle v_i for uplink transmission, and $h_{i,j}$ denote the channel gain from vehicle v_i to RSU s_j. It is important to note that vehicle v_i can only establish communication with an RSU if the received signal is sufficiently strong. Therefore, we define $S_i = \{s_j | R_{i,j}^u > R\text{th}\}$ as the set of RSUs that vehicle v_i can communicate with, where $R_{i,j}^u = P_i^u h_{i,j}$ denotes the received power on RSU s_j from vehicle v_i, and Rth represents the received power threshold [21]. Similarly, the set of vehicles that can communicate with RSU s_j is denoted as V_j. Clearly, for any pair of vehicle v_i and RSU s_j, if $s_j \notin S_i$, then we have $v_i \notin V_j$. In either case, it implies that $p_{i,j} = 0$, indicating that tasks generated by vehicle v_i cannot be offloaded to RSU s_j.

2) Transmission Delay: The task transmission delay is mainly incurred from vehicle to RSU through the uplink channel. The uplink transmission rate $r_{i,j}^u$ from v_i to s_j can be given by the Shannon capacity [6],

$$r_{i,j}^u = B \log_2 \left(1 + \frac{P_i^u h_{i,j}}{\sigma^2 + I_{i,j}}\right), \tag{1}$$

where B is the channel bandwidth, σ^2 is the noise power and $I_{i,j}$ is the interference from other vehicles at the side of RSU s_j for v_i. Due to the limited transmission power of vehicles, the interference $I_{i,j}$ mainly depends on other vehicles in V_j [6], [21], and can be expressed as

$$I_{i,j} = \sum_{v_{i'} \in V_j \setminus \{v_i\}} P_{i'}^u h_{i',j}. \tag{2}$$

Then the average transmission delay of a task from v_i to s_j can be calculated as

$$D_{i,j}^T = \frac{\zeta_i}{r_{i,j}^v}. \tag{3}$$

Following many studies [22], [23] where the result's size after task processing is generally much smaller than its data size before processing, we ignore the transmission delay for sending the task results from RSUs back to vehicles for simplicity. Nevertheless, the following analyses and the proposed solutions are still applicable with these taken into consideration. For example, downlink transmission delay can be reflected by additional transmission delay that changes (3).

3) Transmission Energy Consumption of RSUs: The transmission energy consumption of an RSU is used for transmitting the output data of after executing a task. Suppose that RSU s_j operates at a fixed transmission power P_j^d. The downlink transmission rate $r_{i,j}^d$ between RSU s_j and vehicle v_i can be also obtained by the Shannon capacity.

Let θ_i denote the downlink data size caused by sending one task's result back to v_i. Then, the downlink traffic data size from s_j to v_i in one unit time (e.g. per second) is $\eta_{i,j} = p_{i,j}\phi_i\theta_i$. Therefore, the total transmission energy consumption of s_j in unit time is

$$E_j^T = \sum_{v_i \in V_j} \frac{P_j^d \eta_{i,j}}{r_{i,j}^d} = \sum_{v_i \in V_j} \frac{P_j^d p_{i,j}\phi_i\theta_i}{r_{i,j}^d}. \tag{4}$$

C. Computation Model

1) Computation Delay: Given the offloading decision $p_{i,j}$, $\forall v_i \in V, s_j \in S$, the required computation capacity on s_j posed by tasks from v_i is $\xi_{i,j} = p_{i,j}\phi_i\delta_i$. Accordingly, the total required computation capacity of RSU s_j is given by

$$\lambda_j = \sum_{v_i \in V_j} \xi_{i,j} = \sum_{v_i \in V_j} p_{i,j}\phi_i\delta_i. \tag{5}$$

As tasks' arrival on each vehicle $v_i \in V$ follows a Poisson process, according to the splitting property of Poisson processes [24], the arrival process on each RSU will also follow the Poisson process. Then the computation delay at each RSU can be modeled as an M/GI/1-PS system [25]. So the expected computation delay of one task from v_i on s_j is

$$D_{i,j}^C = \frac{\delta_i}{\mu_j - \lambda_j}. \tag{6}$$

2) Computation Energy Consumption: The computation energy consumption of an RSU depends on the required number of CPU cycles for performing tasks. Therefore, the total computation energy consumption of s_j in one unit time is

$$E_j^C = \kappa_j\lambda_j = \kappa_j \sum_{v_i \in V_j} p_{i,j}\phi_i\delta_i, \tag{7}$$

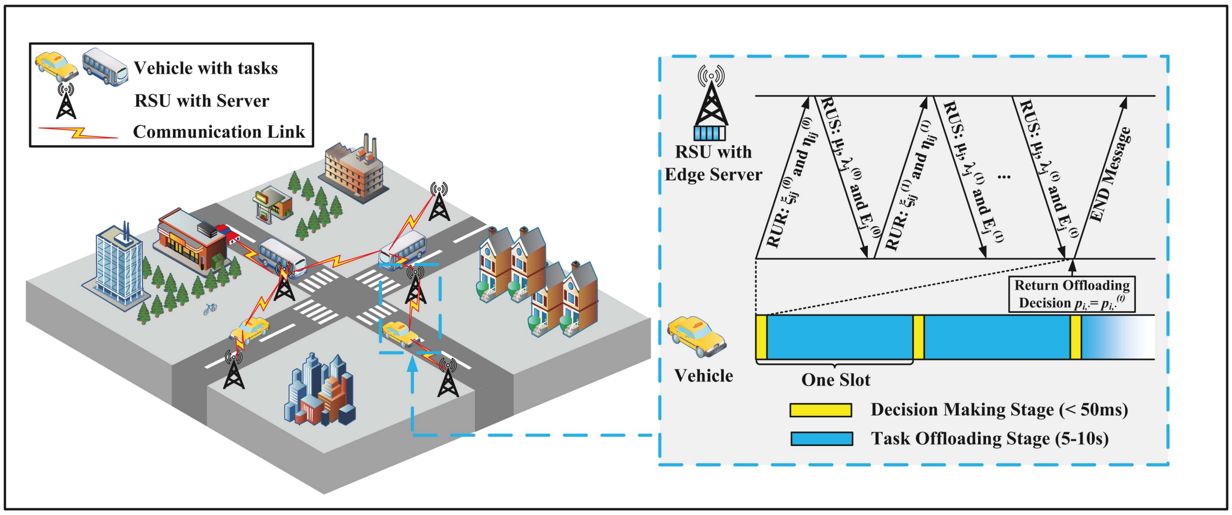

Vehicle with tasks
RSU with Server
Communication Link
RSU with Edge Server
RUR: $\xi_{ij}^{(0)}$ and $\eta_{ij}^{(0)}$
RUS: μ_j, $\lambda_j^{(0)}$ and $E_j^{(0)}$
RUR: $\xi_{ij}^{(1)}$ and $\eta_{ij}^{(1)}$
RUS: μ_j, $\lambda_j^{(1)}$ and $E_j^{(1)}$
...
RUS: μ_j, $\lambda_j^{(t)}$ and $E_j^{(t)}$
END Message
Return Offloading Decision $p_{i,\cdot} = p_{i,\cdot}^{(t)}$
Vehicle
One Slot
Decision Making Stage (< 50ms)
Task Offloading Stage (5-10s)

Fig. 1. Illustration of the workflow.

where κ_j is the energy consumption for executing one CPU cycle on s_j.

III. System Workflow and Problem Formulation

A. System Workflow

The workflow of our solution is depicted in Fig. 1. The system timeline is segmented into a series of fixed-length slots, with a shorter slot chosen to more effectively handle task arrival dynamics compared to a longer slot [26]. By employing efficient algorithms, which we elaborate on in the next section, our solution update offloading decisions every few seconds. This time interval is significantly smaller than the typical assumption made by existing offloading algorithms, which usually operate within time slots ranging from 1 to 5 minutes [27]. Each slot is further divided into two stages: the offloading decision-making stage, responsible for determining task offloading during the slot, and the task offloading stage, which carries out the decided-offloading tasks. We primarily concentrate on the decision-making stage, as it represents the main technical challenges of our approach.

Given the absence of powerful centralized servers, our decision-making stage operates in a distributed manner, enabling vehicles to independently make offloading decisions based on their local information. This decentralized approach maximizes the utilization of computing resources on individual vehicles and reduces the time complexity of the problem, resulting in prompt decision-making. However, since each vehicle possesses only local information, the initial collective offloading decision made by all vehicles may not guarantee satisfactory global performance or comply with resource constraints imposed on certain RSUs. To tackle this challenge, we incorporate feedback from RSUs, enabling vehicles to make appropriate adjustments to their offloading decisions. The following provides a detailed elaboration of this process.

RSU side: Each RSU s_j receives resource-utilization-request (RUR) messages from each vehicle v_i. An RUR message from v_i includes the required computing capacity on s_j for v_i, denoted as $\xi_{i,j}$, as well as the output data size after the execution of tasks offloaded from the vehicle. Based on this information, RSU s_j calculates the total required computing overhead, denoted as λ_j, and determines the total energy consumption necessary for transmitting the output results of all assigned tasks, denoted as E_j. Simultaneously, s_j measures the received power $R_{i,j}^u$ from each v_i and estimates the uplink transmission rate from v_i to s_j. Once this information is obtained, RSU s_j broadcasts its resource-utilization-status (RUS) messages. These messages contain temporary results such as $\lambda_j, E_j, r_{i,j}^u, \mu_j$, and the energy threshold $E_j^{\max}$ of the RSU. It is important to note that λ_j and E_j are not the final computation capacity and energy consumption required on s_j but represent interim values based on the current offloading decisions made by the vehicles.

Vehicle side: Each vehicle v_i determines its offloading decision, represented by the offloading probability to any RSU s_j, denoted as $p_{i,j}$. Initially, a preliminary decision $p_{i,j}, \forall s_j \in S$ is made, and based on this decision, the required information is calculated and included in the resource-utilization-request (RUR) message, which is sent to each RSU s_j. Subsequently, the vehicle awaits the broadcast of resource-utilization-status (RUS) messages from the RSUs, upon which it adjusts its offloading decision accordingly. This iterative process continues until the global performance converges to an optimal or near-optimal state, as we will demonstrate later. Once this state is reached, vehicle v_i sends END messages to the RSUs and proceeds to the task offloading stage. Our simulation results in Section V demonstrate that only a small number (e.g., 5) of iterations are required to achieve efficient offloading decisions for vehicles.

Detailed Algorithms: Our solution incorporates two types of algorithms: one on the RSU side that processes information from RUR messages and generates the necessary outputs for RUS messages, and another on the vehicle side that determines offloading decisions. In the following sections, we will first formulate the problem. Subsequently, we propose algorithms on the RSU side and the vehicle side, respectively (Section IV).

B. Problem Formulation

We make offloading decisions for each vehicle every time slot according to current network traffic and resource conditions. Many applications places a requirement on the task response time. Specifically, the response time of one task from vehicle v_i on RSU s_j consists of its computation delay and transmission delay:

$$D_{i,j} = D^C_{i,j} + D^T_{i,j} = \frac{\delta_i}{\mu_j - \lambda_j} + \frac{\zeta_i}{r^v_{i,j}}. \tag{8}$$

Then, the average response time of all tasks in the system is expressed as

$$\begin{aligned} D(\boldsymbol{p}) &= \frac{1}{\Phi}\sum_{s_j\in S}\sum_{v_i\in V} p_{i,j}\phi_i D_{i,j} \\ &= \frac{1}{\Phi}\sum_{s_j\in S}\left(\frac{\sum_{v_i\in V} p_{i,j}\phi_i\delta_i}{\mu_j - \sum_{v_i\in V} p_{i,j}\phi_i\delta_i} + \sum_{v_i\in V}\frac{p_{i,j}\phi_i\zeta_i}{r^v_{i,j}}\right), \end{aligned} \tag{9}$$

where $\boldsymbol{p} = \{p_{i,j}\}_{v_i\in V, s_j\in S}$ denotes vehicles' decisions in the whole system.

The energy consumption of an RSU consists of its transmission energy consumption and computation energy consumption. Thus, the total energy consumption of RSU s_j in unit time is given by

$$\begin{aligned} E_j = E^T_j + E^C_j &= \sum_{v_i\in V_j}\frac{P^d_j p_{i,j}\phi_i\theta_i}{r^d_{i,j}} + \kappa_j\sum_{v_i\in V_j} p_{i,j}\phi_i\delta_i \\ &= \sum_{v_i\in V_j} p_{i,j}\phi_i\left(\frac{P^d_j\theta_i}{r^d_{i,j}} + \kappa_j\delta_i\right). \end{aligned} \tag{10}$$

Our problem is to making offloading decisions $\boldsymbol{p}_{i,\cdot} = \{p_{i,j}\}_{s_j\in S}$ for each vehicle $v_i \in V$ given the computation capacity and energy constraints of each RSU so as to minimize the average response time of tasks in the system. We formulate our task offloading problem as follows:

$$(\mathbf{P1}):\ \min D(\boldsymbol{p}) \tag{11a}$$
$$\text{s.t.}\quad \lambda_j < \mu_j, \qquad \forall s_j \in S \tag{11b}$$
$$E_j \le E^{\max}_j, \qquad \forall v_i \in V \tag{11c}$$
$$\sum_{s_j\in S_i} p_{i,j} = 1, \qquad \forall v_i \in V \tag{11d}$$
$$0 \le p_{i,j} \le 1, \qquad \forall v_i \in V, s_j \in S. \tag{11e}$$

The first set of inequalities (11b) means that the total computation requirement on an RSU should not exceed its computation capacity. The second set of inequalities (11c) represents that the energy consumption of each RSU in unit time does not exceed an upper limit $E^{\max}_j$. The third set of (11d) ensures that each task will be offloaded to one and only one RSU. Our objective is to minimize the average response time of all tasks in the system, i.e., $\min D(\boldsymbol{p})$.

IV. Algorithm Description

In this section, we first convert the original problem $\mathbf{P1}$ by leveraging exterior-point method (Section IV-A). Then we propose a fully distributed algorithm for task offloading (FDTO), which consists of two parts. One is for RSUs, called FDTO-S (Section IV-B), and the other is for vehicles, called FDTO-V (Section IV-C). We prove that the convergence for FDTO algorithm (Section IV-D).

A. An Exterior-Point Based Optimization Problem

In order to facilitate the description of our distributed algorithms and analyses, we first leverage exterior-point method to transform the constraints of $\mathbf{P1}$ to its optimization objective, such that we can deal with the problem whose solution may exceed the feasible region. Specifically, we construct two constraints (11b) and (11c) as an penalty function

$$\begin{aligned} N(\boldsymbol{p}) = &\sum_{s_j\in S}(\max\{0, \lambda_j - \mu_j + \epsilon\})^2 \\ &+ \sum_{s_j\in S}(\max\{0, E_j - E^{\max}_j\})^2, \end{aligned} \tag{12}$$

where ϵ is an arbitrarily small positive constant. This constant guarantees that the solution will not exceed the boundary of the feasible region of $\mathbf{P1}$. We then add the penalty function $N(\boldsymbol{p})$ to the objective function as

$$F(\boldsymbol{p}) = D(\boldsymbol{p}) + L \cdot N(\boldsymbol{p}), \tag{13}$$

where L is a given large enough positive constant, called the penalty factor. Thus, we convert the original problem to the following optimization problem $\mathbf{P2}$.

$$(\mathbf{P2}):\ \min F(\boldsymbol{p}) \tag{14a}$$
$$\text{s.t.}\quad \sum_{s_j\in S_i} p_{i,j} = 1, \qquad \forall v_i \in V \tag{14b}$$
$$0 \le p_{i,j} \le 1, \qquad \forall v_i \in V, s_j \in S. \tag{14c}$$

Because of the existence of L, if $\boldsymbol{p}$ is chosen such that $\lambda_j \le \mu_j - \epsilon$ or $E_j \le E^{\max}_j$ is not satisfied for an arbitrary RSU s_j, the value of $F(\boldsymbol{p})$ will be extremely large. Therefore, we can find a optimal solution of $\mathbf{P1}$ by solving $\mathbf{P2}$ with a large enough L.

The main challenge to derive the optimal solution of $\mathbf{P2}$ is the incompleteness of information for vehicles in the distributed scenario. It is difficult for a single vehicle to consider the computation capacity and energy constraint on an RSU, while making delay-optimal decision, because it does not have full knowledge of other vehicles' decisions. To address this challenge, we design the FDTO algorithms, such that each vehicle only needs to consider the resource information of its neighboring RSUs.

B. RSU Side Algorithm (FDTO-S)

In this section, we propose the RSU side algorithm called FDTO-S. In the decision making stage, RSUs collect RUR messages from vehicles in each iteration, estimate their required computation capacity, and broadcast RUS messages. Vectors $\boldsymbol{\xi}^{(t)}_{\cdot,j} = \{\xi^{(t)}_{i,j} | v_i \in V_j\}$ and $\boldsymbol{\eta}^{(t)}_{\cdot,j} = \{\eta^{(t)}_{i,j} | v_i \in V_j\}$ represent the

Algorithm 1: FDTO-S on RSU s_j.

```
1: $t = 0$
2: while true do
3:   Receiving RUR/END messages to get $\boldsymbol{\xi}_{\cdot,j}^{(t)}$ and $\boldsymbol{\eta}_{\cdot,j}^{(t)}$
4:   Measure the received power from vehicles
5:   $\lambda_j^{(t)} = \sum_{v_i \in V_j} \xi_{i,j}^{(t)}$
6:   $E_j^{(t)} = \sum_{v_i \in V_j} \left( \frac{P_j^d \eta_{i,j}^{(t)}}{r_{i,j}^d} + \kappa_j \xi_{i,j}^{(t)} \right)$
7:   for each $v_i \in V_j$ do
8:     $r_{i,j}^u = B \log_2 \left( 1 + \frac{R_{i,j}^u}{\sigma^2 + \sum_{v_{i'} \in V_j \setminus \{v_i\}} R_{i',j}^u} \right)$
9:   Broadcast RUS messages containing $\mu_j$, $E_j^{\max}$, $r_{i,j}^u$,
     $\lambda_j^{(t)}$ and $E_j^{(t)}$
10:  $t = t + 1$
```

Algorithm 2: FDTO-V on Vehicle v_i.

```
Input: RSU set $S_i$, threshold $\psi$, step size $\beta$
Output: Final offloading decision $\boldsymbol{p}_{i,\cdot}$
1: for each $s_j \in S_i$ do
2:   Initial offloading decision $p_{i,j}^{(0)} = \frac{1}{|S_i|}$
3:   Send RUR containing $\xi_{i,j}^{(0)}$ and $\eta_{i,j}^{(0)}$ to RSU $s_j$
4: $t = 0$
5: while true do
6:   Receive $\boldsymbol{\mu}_{i,\cdot}$, $\boldsymbol{E}_{i,\cdot}^{\max}$, $\boldsymbol{\lambda}_{i,\cdot}^{(t)}$ and $\boldsymbol{E}_{i,\cdot}^{(t)}$ from RSUs
7:   Calculate $D_i^{(t)}$ by (15)
8:   if $t \geq 1$ and Conditions (16) are satisfied then
9:     for each $s_j \in S_i$ do
10:      Send END message to $s_j$
11:      $\boldsymbol{p}_{i,\cdot} = \boldsymbol{p}_{i,\cdot}^{(t)}$
12:    return $\boldsymbol{p}_{i,\cdot}$
13:  Obtain decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ by FDTO-VG or FDTO-VX
14:  for each $s_j \in S_i$ do
15:    $\xi_{i,j}^{(t+1)} = p_{i,j}^{(t+1)} \phi_i \delta_i$
16:    $\eta_{i,j}^{(t+1)} = p_{i,j}^{(t+1)} \phi_i \theta_i$
17:    Send RUR containing $\xi_{i,j}^{(t+1)}$ and $\eta_{i,j}^{(t+1)}$ to RSU $s_j$
18:  $t = t + 1$
```

received RURs on RSU s_j at iteration t. The required computation capacity and the energy consumption of s_j at iteration t can be estimated by $\lambda_j^{(t)} = \sum_{v_i \in V_j} \xi_{i,j}^{(t)}$ and $E_j^{(t)} = \sum_{v_i \in V_j} (\frac{P_j^d \eta_{i,j}^{(t)}}{r_{i,j}^d} + \kappa_j \xi_{i,j}^{(t)})$, respectively. Then s_j broadcasts its RUS messages containing μ_j, $\lambda_j^{(t)}$ and $E_j^{(t)}$ to vehicles in V_j. Note that if s_j receives an END message from v_i in iteration t, it means that v_i has obtain its final decision and begins to offload tasks, then s_j will set $\xi_{i,j}^{(t')} = \xi_{i,j}^{(t)}$ and $\eta_{i,j}^{(t')} = \eta_{i,j}^{(t)}$, $\forall t' > t$, in the subsequent iterations. The FDTO-S algorithm is described in Algorithm 1.

C. Vehicle Side Algorithm (FDTO-V)

We propose the vehicle side algorithm called FDTO-V. In the following, we first introduce the algorithm initialization and the termination condition. We then present a natural algorithm, called FDTO-VG, with very simple computation. To reduce the number of iterations for convergence, we further present a convex optimization based algorithm (FDTO-VX) for decision making.

1) Algorithm Initialization and Termination: At the beginning, vehicle v_i assigns the uniform offloading probabilities as the initial values, i.e., $p_{i,j}^{(0)} = 1/|S_i|$, $\forall s_j \in S_i$. Then it sends the initial RUR messages $\xi_{i,j}^{(0)} = p_{i,j}^{(0)} \phi_i \delta_i$ and $\eta_{i,j}^{(0)} = p_{i,j}^{(0)} \phi_i \theta_i$ to each RSU $s_j \in S_i$.

At the end of iteration t, vehicle v_i receives the RUS message containing μ_j, $E_j^{\max}$, $\lambda_j^{(t)}$ and $E_j^{(t)}$ from each RSU $s_j \in S$. Then it has the knowledge of the maximum computation capacities $\boldsymbol{\mu}_{i,\cdot} = \{\mu_j | s_j \in S_i\}$, the energy consumption threshold $\boldsymbol{E}_{i,\cdot}^{\max} = \{E_j^{\max} | s_j \in S_i\}$, the estimates of the required computation capacities $\boldsymbol{\lambda}_{i,\cdot}^{(t)} = \{\lambda_j^{(t)} | s_j \in S_i\}$, the energy consumption $\boldsymbol{E}_{i,\cdot}^{(t)} = \{E_j^{(t)} | s_j \in S_i\}$ and the uplink transmission rate $\boldsymbol{r}_{i,\cdot}^{v} = \{r_{i,j}^{v} | s_j \in S_i\}$. The average response time of tasks from v_i in iteration t is given by

$$D_i^{(t)} = \sum_{s_j \in S_i} p_{i,j}^{(t)} D_{i,j}^{(t)} = \sum_{s_j \in S_i} p_{i,j}^{(t)} \left(\frac{\delta_i}{\mu_j - \lambda_j^{(t)}} + \frac{\zeta_i}{r_{i,j}^{v}} \right). \tag{15}$$

We define a small positive threshold ψ, which determines the required accuracy before algorithm termination. Then v_i detects the following three conditions:

$$\lambda_j^{(t)} \leq \mu_j - \epsilon, \forall s_j \in S_i; \tag{16a}$$

$$E_j^{(t)} \leq E_j^{\max}, \forall s_j \in S_i; \tag{16b}$$

$$D_i^{(t)} < (1 + \psi) D_i^{(t-1)}. \tag{16c}$$

Conditions (16a) and (16b) represent that as long as the estimates of the required computation capacity or energy consumption on an arbitrary RSU $s_j \in S_i$ exceed its limits, v_i will continue a new iteration. Condition (16c) represents that the performance difference between two successive iterations is small.

If the above three conditions are satisfied, the iteration will be terminated. v_i uses $\boldsymbol{p}_{i,\cdot}^{(t)}$ as its final decision and sends END messages to RSUs. Otherwise, v_i makes a new offloading decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ of iteration $t + 1$. In Sections IV-C2 and IV-C4, we will propose two solutions about how vehicles obtain the new decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ (Line 13 in Algorithm 2). Then v_i sends an RUR message containing $\xi_{i,j}^{(t+1)} = p_{i,j}^{(t+1)} \phi_i \delta_i$ and $\eta_{i,j}^{(t+1)} = p_{i,j}^{(t+1)} \phi_i \theta_i$ to each RSU $s_j \in S_i$. The FDTO-V algorithm is formally described in Algorithm 2.

2) Greedy Based Algorithm (FDTO-VG): The idea of the vehicle side FDTO-VG algorithm is to simply increase the offloading probability of the optimal RSU and then greedily decreases the probabilities of other RSUs, which will reduce the average response time. From (5) and (13), the objective function of iteration t is regarded as the following $M \times N$-variables continuous function of $\boldsymbol{p}^{(t)}$:

$$\begin{aligned} &F(\boldsymbol{p}^{(t)}) \\ &= D(\boldsymbol{p}^{(t)}) + L \cdot N(\boldsymbol{p}^{(t)}) \\ &= \frac{1}{\Phi} \sum_{s_j \in S} \left(\frac{\sum_{v_i \in V} p_{i,j}^{(t)} \phi_i \delta_i}{\mu_j - \sum_{v_i \in V} p_{i,j}^{(t)} \phi_i \delta_i} + \sum_{v_i \in V} \frac{p_{i,j}^{(t)} \phi_i \zeta_i}{r_{i,j}} \right) \\ &\quad + L \cdot \sum_{s_j \in S} \left[\left(\max \left\{ 0, \sum_{v_i \in V_j} p_{i,j}^{(t)} \phi_i \delta_i - \mu_j + \epsilon \right\} \right)^2 \right. \\ &\quad \left. + \left(\max \left\{ 0, \sum_{v_i \in V_j} p_{i,j}^{(t)} \phi_i \left(\frac{P_j^d \theta_i}{r_{i,j}^d} + \kappa_j \delta_i \right) - E_j^{\max} \right\} \right)^2 \right]. \end{aligned} \tag{17}$$

We compute the partial derivative of $F(\boldsymbol{p}^{(t)})$ with respect to $p_{i,j}^{(t)}$ as

$$\begin{aligned} \frac{\partial F(\boldsymbol{p}^{(t)})}{\partial p_{i,j}^{(t)}} &= \frac{1}{\Phi} \left(\frac{\mu_j \phi_i \delta_i}{(\mu_j - \lambda_j^{(t)})^2} + \frac{\phi_i \zeta_i}{r_{i,j}} \right) \\ &\quad + 2L \cdot \left[\max\{0, \phi_i \delta_i (\lambda_j^{(t)} - \mu_j + \epsilon)\} \right. \\ &\quad \left. + \max\left\{ 0, \phi_i \left(\frac{P_j^d \theta_i}{r_{i,j}^d} + \kappa_j \delta_i \right) (E_j^{(t)} - E_j^{\max}) \right\} \right] \\ &= \frac{\phi_i}{\Phi} \Delta_{i,j}^{(t)}, \end{aligned} \tag{18}$$

where

$$\begin{aligned} \Delta_{i,j}^{(t)} &\triangleq \frac{\mu_j \delta_i}{(\mu_j - \lambda_j^{(t)})^2} + \frac{\zeta_i}{r_{i,j}^u} \\ &\quad + 2L\Phi \left[\max\{0, \delta_i (\lambda_j^{(t)} - \mu_j + \epsilon)\} \right. \\ &\quad \left. + \max\left\{ 0, \left(\frac{P_j^d \theta_i}{r_{i,j}^d} + \kappa_j \delta_i \right) (E_j^{(t)} - E_j^{\max}) \right\} \right] \end{aligned} \tag{19}$$

is defined as the *Preference Index* of RSU s_j at iteration t on vehicle v_i. With the smaller value of $\Delta_{i,j}^{(t)}$, v_i tends to assign a larger offloading probability for s_j in the next iteration (the reason for adopting $\Delta_{i,j}^{(t)}$ as a preference indicator of s_j on v_i will be demonstrated in Section IV-D). Note that as long as one of the inequalities $\lambda_j^{(t)} > \mu_j - \epsilon$, $E_i^{(t)} > E_i^{\max}$ and $E_j^{(t)} > E_j^{\max}$ is satisfied for s_j, $\Delta_{i,j}^{(t)}$ will be extremely large, which means that s_j is overloaded and v_i will decrease the probability $p_{i,j}^{(t+1)}$ in the next iteration.

Let s_{j^*} be the RSU whose $\Delta_{i,j^*}^{(t)}$ is minimum among all RSUs in S_i. Then, we derive a new offloading decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ for v_i as follows

$$\begin{cases} p_{i,j}^{(t+1)} = (1-\beta) p_{i,j}^{(t)}, & s_j \in S_i, j \neq j^* \\ p_{i,j^*}^{(t+1)} = p_{i,j^*}^{(t)} + \beta \sum_{j \neq j^*} p_{i,j}^{(t)}, & s_j \in S_i, \end{cases} \tag{20}$$

where $\beta \in (0, 1]$ is a parameter named *step size*. This parameter can help to 1) ensure that the transformation (20) will converge by iterations; and 2) determine the convergence rate. We will elaborate on this in Section IV-D.

3) An Example of FDTO-VG: To illustrate the distributed iteration process more intuitively, we give a simple example of our FDTO-VG algorithm, and the diagram is shown in Fig. 2. For the sake of illustration, we assume that all RSUs have the uniform computation capacity, all transmission channels have the uniform coefficient, and all tasks are uniform, such that each vehicle v_i's preference indicator $\Delta_{i,j}$ to each RSU s_j will be determined only by the task arrival rate λ_j on s_j. We set β=0.5 in this example.

In Fig. 2, each vehicle can offload tasks to its neighboring RSUs. For example, the tasks of v_2 can be offloaded to s_1, s_2 and s_3, while those of v_1 can only be offloaded to s_1. The task arrival rates of v_1, v_2, v_3 and v_4 are 16, 24, 8 and 30 tasks/s, respectively. At iteration 0, each vehicle offloads the tasks to its neighboring RSUs uniformly. The estimates of task arrival rates on s_1, s_2 and s_3 are 24, 12 and 42 tasks/s, respectively. Then each vehicle updates its probabilities based on its preference indicators to RSUs. For example, from the view of v_2, it has $\Delta_{2,2}^{(0)} < \Delta_{2,1}^{(0)} < \Delta_{2,3}^{(0)}$, so it reduces the offloading probabilities of s_1 and s_3 as $p_{2,1}^{(1)} = 0.5 \times p_{2,1}^{(0)}$ and $p_{2,3}^{(1)} = 0.5 \times p_{2,3}^{(0)}$, respectively, and increases the probability of s_2 as $p_{2,2}^{(1)} = p_{2,2}^{(0)} + 0.5 \times (p_{2,1}^{(0)} + p_{2,3}^{(0)})$. As a result, the estimates of task arrival rates on s_1, s_2 and s_3 will be updated to 20, 22 and 36 tasks/s, respectively, at iteration 1. Finally, the algorithm will converge at iteration 5. Note that convergence does not mean that $\Delta_{i,1}^{(t)} = \Delta_{i,2}^{(t)} = \ldots = \Delta_{i,m}^{(t)}$ for each v_i. For example, considering that $\Delta_{2,1}^{(5)} = \Delta_{2,2}^{(5)} < \Delta_{2,3}^{(5)}$, v_2 is not able to increase probabilities to s_1 and s_2, because $p_{2,3}^{(5)} = 0$. Otherwise, the total probabilities of all RSUs for v_2 will exceed 1.

4) Convex Optimization Based Algorithm (FDTO-VX): In FDTO-VG, the adjustment of offloading decision in each iteration is not refined enough, which requires more iterations for convergence. In this section, we design another vehicle side algorithm which is more precise and converges faster—convex optimization based algorithm (FDTO-VX).

After a vehicle v_i receives $\boldsymbol{\mu}_{i,\cdot}$, $\boldsymbol{\lambda}_{i,\cdot}^{(t)}$ and $\boldsymbol{E}_{i,\cdot}^{(t)}$ from RSUs in iteration t, it supposes that other vehicles' offloading decisions are static (actually, a vehicle cannot obtain the offloading decisions of others in our framework). Then, vehicle v_i adopts $\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)} = \{\hat{p}_{i,1}^{(t+1)}, \ldots \hat{p}_{i,m}^{(t+1)}\}$ as its offloading decision at iteration $t+1$. From the view of v_i, the estimate of the required computation capacity on s_j is $\hat{\lambda}_j^{(t+1)} = \lambda_j^{(t)} - p_{i,j}^{(t)} \phi_i \delta_i + \hat{p}_{i,j}^{(t+1)} \phi_i \delta_i$ at iteration $t+1$. Thus, the estimated average response time of

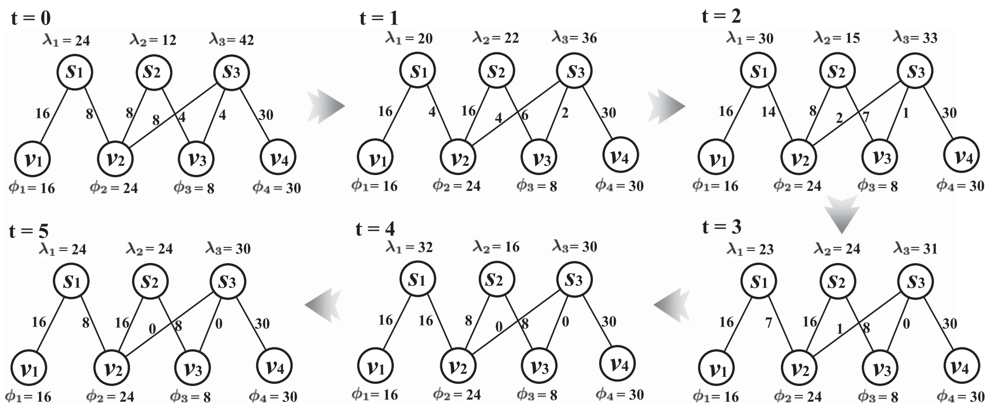


Fig. 2. Example of FDTO-VG.

tasks from vehicle v_i at iteration $t+1$ is given by

$$\begin{aligned}\hat{D}_i^{(t+1)} &= \sum_{s_j\in S_i} \hat{p}_{i,j}^{(t+1)} \hat{D}_{i,j}^{(t+1)} \\ &= \sum_{s_j\in S_i} \hat{p}_{i,j}^{(t+1)} \left(\frac{\zeta_i}{r_{i,j}} + \frac{\delta_i}{\mu_j - \hat{\lambda}_j^{(t+1)}}\right) \\ &= \sum_{s_j\in S_i} \hat{p}_{i,j}^{(t+1)} \left(\frac{\zeta_i}{r_{i,j}} + \frac{\delta_i}{\mu_j - \lambda_j^{(t)} + p_{i,j}^{(t)}\phi_i\delta_i - \hat{p}_{i,j}^{(t+1)}\phi_i\delta_i}\right).\end{aligned} \tag{21}$$

To minimize the average response time of tasks from v_i, $\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)}$ can be obtained by the following problem:

$$(\mathbf{P3}):\ \min \hat{D}_i^{(t+1)} \tag{22a}$$

$$\text{s.t.} \quad \sum_{s_j\in S_i} \Delta_{i,j}^{(t)}\hat{p}_{i,j}^{(t+1)} \le \sum_{s_j\in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)} \tag{22b}$$

$$\sum_{s_j\in S_i} \hat{p}_{i,j}^{(t+1)} = 1 \tag{22c}$$

$$0 \le \hat{p}_{i,j}^{(t+1)} \le 1, \forall s_j \in S_i. \tag{22d}$$

The first inequality (22b) ensures that the adjustment of offloading decision gives a descent direction for average response time, which will be elaborated in Section IV-D. **P3** is a convex problem, and can be solved by deploying the CVXPY package [28] on each vehicle.

$\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)}$ is called a selfish decision for v_i at iteration $t+1$. Note that when all vehicles make selfish decisions, there is no guarantee that the system average response time will decrease. We use a step size β to limit the decision adjustment. The offloading decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ of v_i at iteration $t+1$ is given by the following equation

$$\boldsymbol{p}_{i,\cdot}^{(t+1)} = (1-\beta)\boldsymbol{p}_{i,\cdot}^{(t)} + \beta\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)}, \tag{23}$$

where the step size $\beta \in (0,1]$ has the same effect as that in Section IV-C2.

5) An Example of FDTO-VX: Similar to FDTO-VG, we give an example of FDTO-VX in this section. Given the same network scale, since the value of β in FDTO-VX can be assigned with a larger value than that in FDTO-VG to obtain better convergence (Section V for details), we set β=1 in this example.

In Fig. 3, each vehicle obtains its selfish decision in each iteration by solving **P3**. For example, from the view of v_2, its selfish decision at iteration 1 is $p_{2,1}^{(1)} = 0.25$, $p_{2,2}^{(1)} = 0.75$ and $p_{2,3}^{(1)} = 0$, such that its estimated task arrival rates on s_1, s_2 and s_3 will be 22, 22 and 32 tasks/s, respectively. However, from the view of v_3, its selfish decision at iteration 1 is $p_{3,2}^{(1)} = 1$ and $p_{3,3}^{(1)} = 0$, such that its estimated task arrival rates on s_2 and s_3 are 18 and 38 tasks/s, respectively. At last, the task arrival rates on s_1, s_2 and s_3 is 22, 26 and 30 tasks/s, respectively, at iteration 1. And so on, the algorithm will converge at iteration 2.

D. Convergence Analysis

In this section, we will prove that our algorithms will converge to the global optimal. We denote $\boldsymbol{p}^{(t)} = \{p_{i,j}^{(t)} | \forall v_i \in V, \forall s_j \in S\}$ as the task offloading decisions of all vehicles at iteration t. After vehicles send RUR messages following $\boldsymbol{p}^{(t)}$ and receive RUS messages from RSUs, they execute the FDTO-V algorithm to obtain the offloading decisions $\boldsymbol{p}^{(t+1)}$ at iteration $t+1$. The transformation of $\boldsymbol{p}$ can be expressed as

$$\boldsymbol{p}^{(t+1)} = \mathcal{T}(\boldsymbol{p}^{(t)}). \tag{24}$$

If $\boldsymbol{p}^* = \mathcal{T}(\boldsymbol{p}^*)$, we call that $\boldsymbol{p}^*$ is a fixed point of iterative convergence of (24). If vehicles execute the FDTO-VG algorithm, we denote one iteration transformation of $\boldsymbol{p}$ as $\mathcal{T}_G$. Similarly, if the FDTO-VX algorithm is performed, it is denoted as $\mathcal{T}_X$.

Lemma 1: If $\boldsymbol{p}^{(t)} \neq \boldsymbol{p}^*$, $\mathcal{T}_G(\boldsymbol{p}^{(t)})$ gives a descent direction at $\boldsymbol{p}^{(t)}$ for $F(\boldsymbol{p}^{(t)})$, i.e.,

$$\left\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_G(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \right\rangle < 0,$$

where symbol ∇ is the gradient operator, and $\langle \boldsymbol{a}, \boldsymbol{b}\rangle$ denotes the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

Proof: We compute the partial derivative of $F(\boldsymbol{p}^{(t)})$ with respect to $p_{i,j}^{(t)}$ as $\frac{\partial F(\boldsymbol{p}^{(t)})}{\partial p_{i,j}^{(t)}} = \frac{\phi_i}{\Phi}\Delta_{i,j}^{(t)}$, where $\Delta_{i,j}^{(t)}$ is defined by (19). So,

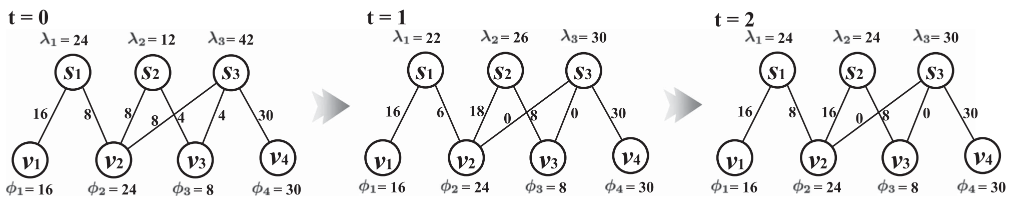


Fig. 3. Example of FDTO-VX.

$$\left\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_G(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \right\rangle = \frac{1}{\Phi} \sum_{p_{i,j}^{(t)} \in \boldsymbol{p}^{(t)}} \phi_i \Delta_{i,j}^{(t)} (p_{i,j}^{(t+1)} - p_{i,j}^{(t)}) = \frac{1}{\Phi} \sum_{v_i \in V} \phi_i \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} (p_{i,j}^{(t+1)} - p_{i,j}^{(t)}). \tag{25}$$

According to Section IV-C2, for vehicle v_i, s_{j^*} is the RSU such that $\Delta_{i,j^*}^{(t)}$ is the minimum of $\Delta_{i,j}^{(t)}$, $s_j \in S_i$. That is, for any RSU $s_j \in S_i$, it follows $\Delta_{i,j^*}^{(t)} \le \Delta_{i,j}^{(t)}$. Then,

$$\Delta_{i,j^*}^{(t)} \sum_{s_j \in S_i, j \ne j^*} p_{i,j}^{(t)} \le \sum_{s_j \in S_i, j \ne j^*} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}. \tag{26}$$

By (20) and (26), we can deduce that

$$\begin{aligned} \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t+1)} &= \Delta_{i,j^*}^{(t)} p_{i,j^*}^{(t+1)} + \sum_{s_j \in S_i, j \ne j^*} \Delta_{i,j}^{(t)} p_{i,j}^{(t+1)} \\ &= \Delta_{i,j^*}^{(t)} \left(p_{i,j^*}^{(t)} + \beta \sum_{\substack{s_j \in S_i, \\ j \ne j^*}} p_{i,j}^{(t)} \right) + \sum_{\substack{s_j \in S_i, \\ j \ne j^*}} \Delta_{i,j}^{(t)} (1-\beta) p_{i,j}^{(t)} \\ &= \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)} + \beta \left(\Delta_{i,j^*}^{(t)} \sum_{\substack{s_j \in S_i, \\ j \ne j^*}} p_{i,j}^{(t)} - \sum_{\substack{s_j \in S_i, \\ j \ne j^*}} \Delta_{i,j}^{(t)} p_{i,j}^{(t)} \right) \\ &\le \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}. \end{aligned} \tag{27}$$

Since $F(\boldsymbol{p}^{(t)})$ is convex and $\boldsymbol{p}^{(t)} \ne \boldsymbol{p}^*$, there exists v_i and $s_j \in S_i$ such that $\Delta_{i,j}^{(t)} > \Delta_{i,j^*}^{(t)}$. So there exists at least one vehicle v_i with $\Delta_{i,j^*}^{(t)} \sum_{s_j \in S_i, j \ne j^*} p_{i,j}^{(t)} < \sum_{s_j \in S_i, j \ne j^*} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}$, such that $\sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t+1)} < \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}$. Hence, $\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_G(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \rangle < 0$.

Lemma 2: If $\boldsymbol{p}^{(t)} \ne \boldsymbol{p}^*$, $\mathcal{T}_X(\boldsymbol{p}^{(t)})$ gives a descent direction at $\boldsymbol{p}^{(t)}$ for $F(\boldsymbol{p}^{(t)})$, i.e.,

$$\left\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_X(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \right\rangle < 0.$$

Proof: Similar to Lemma 1,

$$\left\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_X(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \right\rangle = \frac{1}{\Phi} \sum_{v_i \in V} \phi_i \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} (p_{i,j}^{(t+1)} - p_{i,j}^{(t)}). \tag{28}$$

By (22b) and (23),

$$\begin{aligned} \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t+1)} &= \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} ((1-\beta) p_{i,j}^{(t)} + \beta \hat{p}_{i,j}^{(t+1)}) \\ &\le \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} ((1-\beta) p_{i,j}^{(t)} + \beta p_{i,j}^{(t)}) \\ &= \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}. \end{aligned} \tag{29}$$

As (17) is convex, if $\sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t+1)} = \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}$, we can deduce that $\mathcal{T}_X(\boldsymbol{p}^{(t)}) = \boldsymbol{p}^{(t)}$, and $\boldsymbol{p}^{(t)} = \boldsymbol{p}^*$. Hence, $\sum_{s_j \in S_i} \Delta_{i,j}^{(t)} \hat{p}_{i,j}^{(t+1)} < \sum_{s_j \in S_i} \Delta_{i,j}^{(t)} p_{i,j}^{(t)}$, and $\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}_X(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \rangle < 0$ also holds.

Lemma 3: For $\mathcal{T}_G$ and $\mathcal{T}_X$, if $\boldsymbol{p}^{(t)} \ne \boldsymbol{p}^*$, there exists $\beta \in (0, 1]$ such that $F(\boldsymbol{p}^{(t+1)}) < F(\boldsymbol{p}^{(t)})$.

Proof: If $\boldsymbol{p}^{(t)} \ne \boldsymbol{p}^*$, according to the definition of the fixed point, $\boldsymbol{p}^{(t+1)} = \mathcal{T}(\boldsymbol{p}^{(t)}) \ne \boldsymbol{p}^{(t)}$. We regard $\boldsymbol{p}^{(t+1)}$ as a variable vector that varies with the value of parameter β, denoted as $\boldsymbol{p}^{(t+1)}(\beta)$. $\boldsymbol{p}^{(t)}$ is a constant vector when $\beta = 0$, which means the offloading decisions of iteration $t + 1$ are same as those of iteration t, i.e., $\boldsymbol{p}^{(t+1)}(0) = \boldsymbol{p}^{(t)}$. By the definition of $F(\boldsymbol{p}^{(t)})$ in (17), $F(\boldsymbol{p}^{(t+1)})$ is differentiable for each component of $\boldsymbol{p}^{(t+1)}$. Since both (20) and (23) are linear transformations, the components of $\boldsymbol{p}^{(t+1)}$ can be differentiable on β. Thus, $F(\boldsymbol{p}^{(t+1)})$ is differentiable on β, too. To simplify expression, we define $G(\beta) = F(\boldsymbol{p}^{(t+1)}(\beta))$ as a function of β, and $G(0) = F(\boldsymbol{p}^{(t+1)}(0)) = F(\boldsymbol{p}^{(t)})$.

Assume that $\forall \beta \in (0, 1]$, $G(\beta) \ge G(0)$, i.e., $F(\boldsymbol{p}^{(t+1)}) \ge F(\boldsymbol{p}^{(t)})$. On one hand,

$$G'(0^+) = \lim_{\beta \to 0^+} \frac{G(\beta) - G(0)}{\beta - 0} \ge 0. \tag{30}$$

On the other hand, by chain rule,

$$\begin{aligned} G'_\beta &= \sum_{p_{i,j} \in \boldsymbol{p}^{(t+1)}} \frac{\partial F}{\partial p_{i,j}^{(t+1)}} \cdot \frac{\partial p_{i,j}^{(t+1)}}{\partial \beta} \\ &= \left\langle \nabla F(\boldsymbol{p}^{(t+1)}), (\boldsymbol{p}^{(t+1)})'_\beta \right\rangle. \end{aligned} \tag{31}$$

By Lemmas 1 and 2, for $\beta \to 0^+$,

$$\begin{aligned} G'(0^+) &= G'_\beta|_{\beta \to 0^+} \\ &= \lim_{\beta \to 0^+} \left\langle \nabla F(\boldsymbol{p}^{(t+1)}), \frac{\boldsymbol{p}^{(t+1)}(\beta) - \boldsymbol{p}^{(t+1)}(0)}{\beta} \right\rangle \\ &= \lim_{\beta \to 0^+} \frac{\langle \nabla F(\boldsymbol{p}^{(t)}), \mathcal{T}(\boldsymbol{p}^{(t)}) - \boldsymbol{p}^{(t)} \rangle}{\beta} < 0, \end{aligned} \tag{32}$$

which is a conflict with inequation (30). As a result, the assumption is false. That is, there exists $\beta \in (0, 1]$ such that $F(\boldsymbol{p}^{(t+1)}) < F(\boldsymbol{p}^{(t)})$.

Theorem 1: For $\mathcal{T}_G$ and $\mathcal{T}_X$, suppose that β makes $F(\boldsymbol{p}^{(t+1)}) < F(\boldsymbol{p}^{(t)})$. Then $\boldsymbol{p}^{(t)}$ can converge to $\boldsymbol{p}^*$, i.e., $\lim_{t\to+\infty} \boldsymbol{p}^{(t)} = \boldsymbol{p}^*$. In addition, $\boldsymbol{p}^*$ is the optimal solution of problem ***(P2)***.

Proof: $F(\boldsymbol{p}^{(t)})$ is a monotonically decreasing sequence in t and $F(\boldsymbol{p}^{(t)}) > 0$, so $F(\boldsymbol{p}^{(t)})$ will converge. Suppose that $\lim_{t\to+\infty} \boldsymbol{p}^{(t)} = \boldsymbol{p}', \boldsymbol{p}' \neq \boldsymbol{p}^*$, according to Lemma 3, if $\boldsymbol{p}' \neq \boldsymbol{p}^*$, there exists $\beta \in (0, 1]$ such that $F(\boldsymbol{p}')$ decreases again. So $\lim_{t\to+\infty} \boldsymbol{p}^{(t)} = \boldsymbol{p}^*$. In addition, suppose that there exists $\boldsymbol{p}''$ such that $\mathcal{T}(\boldsymbol{p}'') \neq \boldsymbol{p}''$, and $\boldsymbol{p}''$ is the vector with the minimum value of $F(\boldsymbol{p})$. According to Lemmas 1–3, $\mathcal{T}(\boldsymbol{p}'')$ gives a descent direction at $\boldsymbol{p}''$, and there exists a $\beta \in (0, 1]$ such that $F(\mathcal{T}(\boldsymbol{p}'')) < F(\boldsymbol{p}'')$. It's a contradiction. So, $\boldsymbol{p}^*$ is the optimal solution of problem $\mathbf{P2}$.

E. Discussion

1) Time Complexity Analysis: The original problem **P1** is a mixed integer nonlinear programming (MINLP) problem. Solving **P1** centrally requires a controller to know the resource usage state of m RSUs and to make offloading decisions for all n vehicles, which has a time complexity of $O(H(mn))$. Here, $H(x)$ is the completion time for solving an MINLP problem of scale x, which depends on the method used, such as the branch and bound (BB) algorithm [29] or learning-based method [12], [30]. For game-based distributed schemes, the time complexity is $O(nH(m))$, which is also related to the number of vehicles. In contrast, our FDTO-VG algorithm only requires each vehicle to perform a simple linear calculation (20) in each round to make its own offloading decision, resulting in a time complexity of $O(k)$, where k is the number of iterations. Similarly, our FDTO-VX algorithm requires each vehicle to solve an MINLP problem **P3** of scale m in each round, leading to a time complexity of $O(kH(m))$. Our experimental results in Section V will demonstrate that the number of iterations k is much smaller than the number of vehicles n. Therefore, we can conclude that our FDTO-VG and FDTO-VX algorithms outperform the existing algorithms significantly in terms of time complexity.

2) Impact of β on System Performance: Our simulation results in Section V-C1 show the impact of β on the system performance, such as the number of iterations for convergence and the task response time. We briefly analyze these simulation results and propose the improved algorithm.

The choice of parameter β is crucial for both vehicle side algorithms. To guarantee convergence, the magnitude of β depends on the task arrival rate in the system. When the task arrival rate is much slower than the processing rate of RSUs in the system, a large value of β (e.g., 0.5–1) can speed up the convergence rate for both FDTO-VG and FDTO-VX.

However, we also concern about the situation where the required computation resource is close to the RSUs' capacity. Our simulation results show that the performance of FDTO-VG is not completely satisfied when the offloading request rate is high. On one hand, if a small value of β (e.g., 0.01–0.05) is chosen, the algorithm will achieve a shorter task response time after multiple iterations, because offloading decisions will be slightly adjusted in each iteration. However, it will take a large number of iterations to exchange information between vehicles and the RSUs (4 ms per iteration in our simulation), which leads to a long decision making time. On the other hand, given a large value of β, the algorithm will not converge to a good level, because the adjustment way of FDTO-VG is relatively rough.

On the contrary, if the value of β is properly chosen, FDTO-VX not only needs a fewer number of iterations for convergence than FDTO-VG, but also achieves a smaller task response time. Nevertheless, vehicles take about 10 ms to solve a convex optimization problem in each iteration when running the FDTO-VX algorithm according to our evaluation on the testbed with a core i7-3770 processor, while the computing time of FDTO-VG on vehicles can be ignored because FDTO-VG only involves linear operations.

3) Algorithm Improvement: We combine the advantages of both FDTO-VG and FDTO-VX. Usually, the required computation capacity on an RSU is much less than its maximum computation capacity due to the imbalance of task offloading requests in the system. Though FDTO-VG and FDTO-VX will converge with similar number of iterations in this case, FDTO-VG takes less time for each iteration for its simple computation. In addition, if the requested computation capacity is close to or even exceeds the RSUs' capacity, FDTO-VX will achieve convergence in a fewer number of iterations which leads to less decision making time. We combine FDTO-VG and FDTO-VX into an algorithm FDTO-VC, which is formally described in Algorithm 3. The choice of which algorithm on vehicle v_i depends on the following judgment: if $\frac{\sum_{s_j\in S_i}\lambda_j^{(t)}}{\sum_{s_j\in S_i}\mu_j} \leq \alpha$, v_i runs FDTO-VG at iteration $t+1$; Otherwise, v_i runs FDTO-VX. It is expected to obtain a $\alpha \in [0, 1)$, named *activation boundary*, to achieve satisfactory system performance, i.e., a short decision making time no matter the task arrival rate of vehicles. If vehicles execute the FDTO-VC algorithm, we denote one iteration transformation of $\boldsymbol{p}$ as $\mathcal{T}_C$.

Algorithm 3: FDTO-VC to obtain $\boldsymbol{p}_{i,\cdot}^{(t+1)}$.

```
1: if $\sum_{s_j\in S_i}\rho_j^{(t)}\mu_j / \sum_{s_j\in S_i}\mu_j \leq \alpha$ then
2:    // Execute the FDTO-VG Algorithm
3:    Find $s_{j^*}$ such that $\Delta_{i,j^*}^{(t)} = \min\{\Delta_{i,j}^{(t)}\}_{s_j\in S_i}$
4:    Obtain offloading decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ by (20)
5: else
6:    // Execute the FDTO-VX Algorithm
7:    Solve problem P2 to obtain $\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)}$
8:    Obtain offloading decision $\boldsymbol{p}_{i,\cdot}^{(t+1)}$ by (23)
```

Theorem 2: The iterative transformation of $\boldsymbol{p}$ with $\mathcal{T}_C(\boldsymbol{p})$ can make $F(\boldsymbol{p})$ converge.

Proof: We can be easily obtained the convergence of $\mathcal{T}_C(\boldsymbol{p})$ by the convergence of $\mathcal{T}_G$ and $\mathcal{T}_X$ in Theorem 1.

4) Potential Modeling Inaccuracy in Practice: In Section II, we adopt the queuing theory and the Shannon capacity to model the computation time and the transmission time, respectively, for the sake of theoretical analysis and simulation. However, these models may not capture the dynamic and complex nature of wireless networks, and thus the task completion time in reality may deviate from these mathematical mappings. One possible way to overcome this challenge is to use deep reinforcement learning (DRL) [12], which can build a policy network according to various factors in the actual environment, and address the difficulty of finding the optimal mapping between environment and decisions to minimize the response time. In our future work, we aim to implement FDTO algorithm in a practical system, and apply DRL to FDTO algorithm to update the decision $\hat{\boldsymbol{p}}_{i,\cdot}^{(t+1)}$ at each iteration based on the environment information.

5) Communication Disruption Caused by Mobility: Vehicle mobility in the wireless environment may cause communication disruption between vehicles and RSUs. Our system workflow can be easily extended to solve this problem by applying triggers properly to govern handover management control. For example, when a vehicle v_i leaves the transmission range of an RSU or occurs failure connection in a time slot, it will immediately update its communicated RSU set S_i, and run the vehicle side algorithm FDTO-V to obtain a new offloading decision, rather than waiting for the end of the current slot. The decision making delay is short so that it will not significantly influence with the offloading of new-arrival tasks. Our simulation results in Section V show that as long as only a small percentage of vehicles are involved in communication handover within a time slot, the overall performance has little influence.

V. Numerical Evaluation

A. Simulation Settings

We simulate a typical vehicle network scenario. All simulations are performed on a desktop computer with an Intel(R) Core(TM) i7-3770 CPU running at 3.4 GHz and 8 GB RAM. The operating system is Ubuntu 18.04 LTS. The simulation software is Python 3.7. The settings of some important parameters are listed in Table III.

Scenario: We simulate a 5 km $\times$ 5 km square city area with a total road length of 50 km divided by 50 $\times$ 50 grids [31]. We assume that a road side unit (RSU), i.e., an access point, is deployed in the exact center of each road grid, so totally 500 RSUs are deployed. Each RSU is equipped with an MEC RSU, whose maximum computation capacity and energy constraint are set as 5 GHz and 80 W, respectively.

Channel Mode: To adapt to the simulation scenario, the wireless access technique between vehicles and RSUs is based on IEEE 802.11p standard with 10 MHz channel bandwidth [32].

TABLE III
Settings of Some System Parameters

Parameters	Value
Simulated city	5km×5km [31]
Number of RSUs	500
Number of vehicles	2,000-10,000
Channel bandwidth	10 MHz [32]
Transmission power of Vehicles	100 mW [6]
Noise power	-100 dBm [6]
Request packet length	1554 Bytes [33]
Max. computation capacity of RSUs	5 GHz
Expected No. of CPU cycles per-task	100 M
Expected input data size per-task	0.1 Mb
Transmission power of RSUs	33 dBm
Energy consumption for a CPU cycle	$[10, 50]$ nJ
Energy constraint of each RSU	80 W

The channel gain for calculating the wireless transmission is modeled by the path-loss $h_{i,j} = h_0 d_{i,j}^{-4}$ model [34], where $h_0 = -40$ dB is the path-loss constant and $d_{i,j}$ is the distance between vehicle u_i and RSU s_j. The transmission power of each vehicle is set as $P_i^u = 100$ mW, and the noise power is set as $\sigma^2 = -100$ dBm [6]. In the decision making stage, vehicles and RSUs transmit packets (1554 Bytes [33]) to exchange RUR and RUS messages.

Traffic: We simulate the number of vehicles as 2,000, 4,000, 6,000, 8,000, 10,000 randomly distributed on the road of the simulated city area. They travel at random speeds from 40 km/h to 80 km/h. Each vehicle will pick a random direction when it comes to an intersection. For any vehicle, its task generation follows a Poisson process. Unless otherwise specified, the expected input data size of each task is 0.1 Mb, and the result data size is set to be drawn evenly from 0% to 200% of the task's input size. The expected required number of CPU cycles of one task is 100 M. Consider that the transmission power of RSUs is 33 dBm, the energy consumption for performing one CPU cycle on an RSU is randomly drawn from [10,50] nJ.

B. Benchmarks and Performance Metrics

We choose three offloading algorithms as benchmarks.

Centralized Task Offloading (CTO): A centralized controller has the whole knowledge of the network information and collects the requirements of vehicles, then it solves the problem **P1** to obtain the offloading decisions, aiming to achieve optimal average delay in the system, called *CTO-DO*. CTO-DO can also be regarded as a lower bound of average response time of our problem. Correspondingly, the algorithm that optimizes total energy consumption is called *CTO-EO*. For comparison, we assume that all tasks will be offloaded only to the RSUs for processing.

Multi-armed Bandit (MAB) [18]: A distributed decision solution based on multi-armed bandit which optimizes the task response time with only local information.

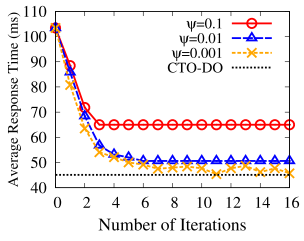


Fig. 4. Average response time vs. iterations with different ψ.

Noncooperative Game based Task Offloading (NGTO): Another distributed task offloading algorithm is based on a noncooperative game [16]. Specifically, vehicles run this algorithm with a round-robin fashion in turn. In each round, a vehicle needs to obtain offloading decisions of other vehicles and executes best-response algorithm based on the collected information. Then it sends its offloading decision to other vehicles immediately. This process will terminate until a Nash equilibrium is reached.

We use the following metrics in our numerical evaluation to demonstrate the efficiency of our solutions.

Average Response Time: After offloading decisions of vehicles are confirmed, we can obtain the average response time of all tasks in the whole system.

Average Energy Consumption: We also compare the average energy consumption among all RSUs in the system with other benchmarks.

Decision Making Time: Decision making time of a solution consists of two parts: the algorithm's running time on devices (vehicles or RSUs) and transmission time among devices in the decision making stage. In MAB and our solutions, the transmission time occur in the data transmission between vehicles and RSUs. In CTO, the decision time occurs when the centralized controller collects the requests from vehicles and sends the decisions back. In NGTO, it occurs in the process that vehicles take turns to send their offloading decisions to others.

C. Simulation Results

1) Parameter Matching: In this section, we evaluate the impact of some parameters on the decision making time of our algorithms, including the threshold ψ, the average task arrival rate of vehicles $\overline{\phi} = \Phi/|U|$, the step size β and the activation boundary α. The selection of different algorithms and parameters results in different number of iterations and time of one iteration, which determine the decision making time. There are 10,000 simulated vehicles in total in the simulations. As a comparison, we assume that there is a centralized controller which has the whole knowledge of the network and solves the problem $\mathbf{P1}$ to provide the optimal solution (CTO-DO). Figs. 4–10 show the changing trend of task average response time as the number of iterations increases. In general, with more iterations, the algorithms will decrease the average response time through efficiently adjusting offloading decisions. But the convergence trend varies with the different parameter values.

Fig. 4 shows the impact of threshold ψ on the convergence performance of FDTO-VX (FDTO-VG has similar trend). In

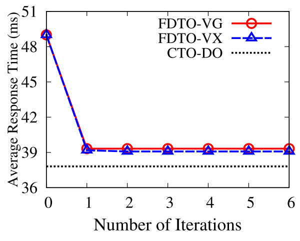


Fig. 5. Average response time vs. iterations with $\overline{\phi} = 0.5$.

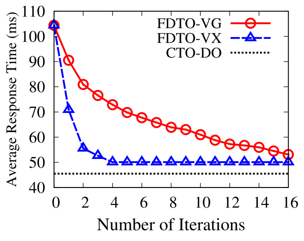


Fig. 6. Average response time vs. iterations with $\overline{\phi} = 1$.

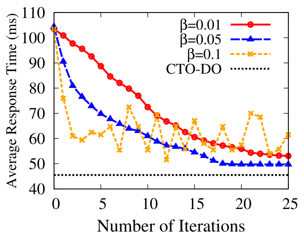


Fig. 7. Average response time vs. iterations using FDTO-VG.

this simulation, we set $\overline{\phi}$=1 task/s and β=0.05. It shows that, if ψ is set as 0.1, the algorithm prematurely converges after 3 iterations, and the average task response time is 65.1 ms, far larger than the optimal result (i.e., 45.5 ms). When ψ is set as 0.01, the algorithm converges after 6 iterations with average response time of 50.3 ms. When ψ is 0.001, the algorithm will not converge even after 16 iterations, since the difference in average task response time of vehicles between two adjacent iterations is rarely be less than such a small threshold. Based on this observation, we experientially set the threshold ψ as 0.01 in all subsequent simulations.

Figs. 5 and 6 show the convergence performance of both vehicle side algorithms when the average task arrival rate of vehicles $\overline{\phi}$ is set as 0.5 and 1 tasks/s, respectively. As shown in Fig. 5, when the task arrival rate is much lower than the RSU's computation capacity, both FDTO-VG and FDTO-VX will converge after only 2 iterations. However, when the task arrival rate is set as 1 tasks/s in Fig. 6, FDTO-VG requires more than 16 iterations to achieve convergence, while FDTO-VX only requires 6 iterations. Therefore, we set $\overline{\phi}$ as 1 task/s in the subsequent simulations.

Figs. 7–9 display the impact of the step size β on the performance of vehicle side algorithms FDTO-VG, FDTO-VX and FDTO-VC, respectively. Fig. 7 shows the convergence performance of FDTO-VG when the value of β is chosen as 0.01, 0.05 and 0.1, respectively. When $\beta = 0.01$, FDTO-VG requires

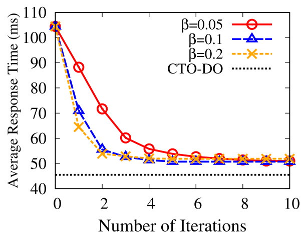


Fig. 8. Average response time vs. iteration using FDTO-VX.

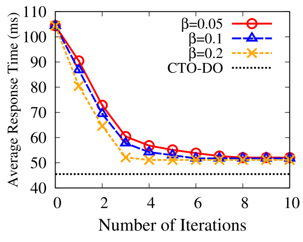


Fig. 9. Average response time vs. iterations using FDTO-VC.

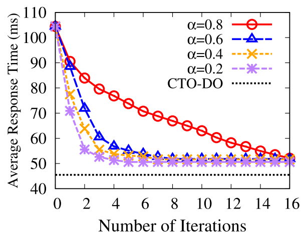


Fig. 10. Average response time vs. iterations using FDTO-VC.

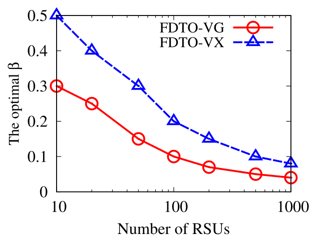


Fig. 11. The optimal β vs. number of RSUs.

more than 25 iterations to achieve convergence. For example, the average response time is 53.1 ms after 25 iteration, while the average response time of CTO-DO is 45.5 ms. When β is set as 0.05, FDTO-VG requires 20 iterations to achieve satisfactory convergence, and its average response time is 49.7 ms after 20 iterations. When β is 0.1, the average response time fluctuates between 50 ms and 70 ms after 2 iterations. It shows that FDTO-VG can converge when the value of β is small, but a small value of β will result in more iterations for convergence. On the other hand, FDTO-VG may not converge given a relatively large β. Fig. 8 shows the changing trend of average response time of FDTO-VX when the value of β is set as 0.05, 0.1 and 0.2, respectively. Specifically, FDTO-VX requires about 9, 5, and 5 iterations to converge when β is 0.05, 0.1 and 0.2, respectively. For example, when β is 0.1, the average response time of FDTO-VX is 50.7 ms after 5 iterations, which is very close to that of CTO-DO. It shows that FDTO-VX will converge faster than FDTO-VG given a larger value of β. Fig. 9 shows the convergence performance of FDTO-VC (α=0.4) when the value of β is set as 0.05, 0.1 and 0.2, respectively. As shown in this figure, FDTO-VC requires about 8, 6, 4 iterations to achieve satisfactory convergence when β is 0.05, 0.1 and 0.2, respectively. It shows that FDTO-VC has similar convergence performance with FDTO-VX.

Fig. 10 shows the impact of the activation boundary α on the performance of FDTO-VC. In this simulation, we specify β=0.05. As shown in this figure, on one hand, when α=0.8, FDTO-VC requires a large number (more than 16) of iterations to achieve convergence. On the other hand, when α is set as 0.2, 0.4 or 0.6, it requires less (about 5-8) iterations for convergence.

Fig. 11 shows the optimal value of β when the number of RSUs increases from 10 to 1000. In this simulation, we set the ratio of the number of vehicles to the number of RSUs as 20, i.e., $\frac{n}{m} = 20$, and the task arrival rate of each vehicle is set as 1 task/s. As shown in Fig. 11, as the number of RSUs increases in magnitude, the optimal β value gradually decreases. For example, when there are only 10 RSUs, the optimal β's of FDTO-VG and FDTO-VX are 0.3 and 0.5, respectively, while with 1000 RSUs, the optimal β's of them are 0.04 and 0.08 respectively. The simulation results show that the optimal β usually depends on the network scale. Generally, in a large-scale network, when a tremendous amount of vehicles make decisions simultaneously, a large step size β can easily cause the decision to converge fast or oscillate near the optimal (as shown in Fig. 7). On the contrary, a small β will contribute to make the decision gradually approach the optimal but with a slow pace.

Table IV summarizes the decision making time of different vehicle side algorithms. When the average task arrival rate is 0.5 tasks/s, all algorithms will converge in 2 iterations. When the average task arrival rate is increased to 1 tasks/s, it becomes more complicated. One iteration takes around 4 ms (i.e., negligible execution time on vehicles/RSUs and 4 ms for transmission) for FDTO-VG and 14 ms (i.e., 10 ms for execution and 4 ms for transmission) for FDTO-VX. As a result, the decision making time cannot be less than 70 ms in both the algorithms. For FDTO-VC, no matter which vehicle executes FDTO-VX, the iteration time is 14 ms. On the contrary, if all vehicles execute FDTO-VG, the iteration time shrinks to 4 ms. We find that when α is set as 0.4, 0.6 or 0.8, the first iteration takes 14 ms and the others take 4 ms, because vehicles' request computation resources are more balanced after the first iteration, and all vehicles will execute FDTO-VG. But when α is set as 0.2, each iteration takes 14 ms, since the boundary is too low such that at least one vehicle will execute FDTO-VX in each iteration. The table shows that when α is set from 0.4-0.6, we can choose a proper value of β such that the decision making time can be less than 50 ms. For example, when α=0.4 and β=0.1, the decision making time is only 34 ms.

2) Performance Comparison With Benchmarks: In this section, we simulate the algorithms in *non-mobility* scenarios, because CTO and NGTO do not fit for the mobile application scenarios. In this and the next section, we use FDTO-VC as our vehicle side algorithm.

TABLE IV
VEHICLE SIDE ALGORITHM COMPARISON

Task Rate ($tasks/s$)	Algorithm		Convergence Performance		One Iteration Time (ms)	Decision Making Time (ms)
			Convergent	No. of Iterations		
$\overline{\phi} = 0.5$	FDTO-VG (any β)		yes	2	4	8
	FDTO-VX (any β)		yes	2	14	28
	FDTO-VC (any β)		yes	2	4	8
$\overline{\phi} = 1$	FDTO-VG	$\beta = 0.01$	yes	37	4	148
		$\beta = 0.05$	yes	20	4	80
		$\beta = 0.1$	no	-	4	-
	FDTO-VX	$\beta = 0.05$	yes	9	14	126
		$\beta = 0.1$	yes	5	14	70
		$\beta = 0.2$	yes	5	14	70
	FDTO-VC ($\alpha = 0.2$)	$\beta = 0.05$	yes	9	14	126
		$\beta = 0.1$	yes	6	14	84
		$\beta = 0.2$	yes	4	14	56
	FDTO-VC ($\alpha = 0.4$)	$\beta = 0.05$	yes	9	4 or 14	46
		$\beta = 0.1$	yes	6	4 or 14	34
		$\beta = 0.2$	yes	6	4 or 14	34
	FDTO-VC ($\alpha = 0.6$)	$\beta = 0.05$	yes	10	4 or 14	50
		$\beta = 0.1$	yes	7	4 or 14	38
		$\beta = 0.2$	yes	6	4 or 14	34
	FDTO-VC ($\alpha = 0.8$)	$\beta = 0.05$	yes	18	4 or 14	82
		$\beta = 0.1$	no	-	4 or 14	-
		$\beta = 0.2$	no	-	4 or 14	-

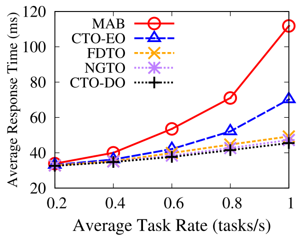


Fig. 12. Average response time vs. task arrival rate.

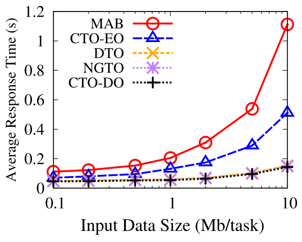


Fig. 13. Average response time vs. input data size.

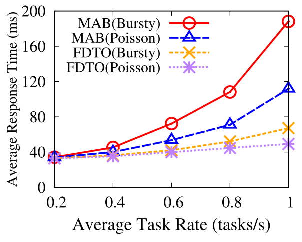


Fig. 14. Average response time vs. task arrival rate.

Fig. 12 shows that the average response time is increasing with higher task arrival rate. When the average task arrival rate of vehicles is low, such as 0.2–0.4 tasks/s, system resource is sufficient so that the average response time of all algorithms is almost the same. But when the average task arrival rate of vehicles is getting higher and system resource becomes insufficient, the average response time of MAB grows rapidly, while our FDTO algorithm gets a little worse performance than CTO-DO and NGTO. For example, when the task arrival rate is 1 tasks/s, the average response time of MAB, CTO-EO, FDTO, NGTO and CTO-DO is 111.8 ms, 70.3 ms, 49.22 ms, 47.24 ms and 45.5 ms, respectively.

In practice, there are different types of applications with different data sizes for their tasks. Fig. 13 shows the average response time as the magnitude of input data size increases from 0.1Mb/task to 10Mb/task. In this simulation, we set the required CPU cycles per-task is constant 100 M. As shown in this figure, task response time increases with the input data size, as it causes the increase of transmission delay. However, our FDTO is much better than MAB and CTO-EO at different input data size per-task. For example, when the input data size is 10Mb/task, the average response time of MAB, CTO-EO, FDTO, NGTO and CTO-DO is 1112.2 ms, 511.2 ms, 153.4 ms, 148.1 ms and 143.2 ms, respectively.

As the task arrival pattern in the real-world system may not follow the assumed Poisson process, we also simulate our algorithm with different task arrival patterns. Fig. 14 shows the average response time of MAB and FDTO with task Poisson arrival and Bursty arrival, respectively. As shown in this figure, the average response time of MAB and FDTO with Bursty arrival is higher than that with Poisson arrival. This is because when the bursts occur, there will be more tasks queuing, resulting in higher computation delay. However, our FDTO algorithm can reduce average response time compared to MAB with both the two task arrival patterns. For example, when the task arrival rate is 1 tasks/s, the average response time of MAB (Bursty), MAB

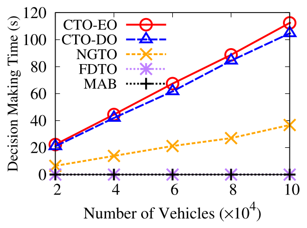


Fig. 15. Decision making time vs. number of vehicles.

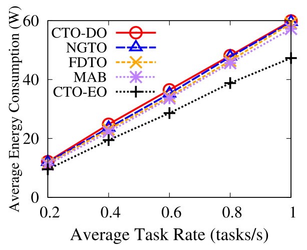


Fig. 16. Average energy consumption vs. task arrival rate.

(Poisson), FDTO (Bursty) and FDTO (Poisson) is 188.24 ms, 111.8 ms, 67.23 ms and 49.22 ms, respectively.

Fig. 15 shows the decision making time of different algorithms with the increasing number of vehicles. As shown in this figure, the decision making time of NGTO, CTO-DO and CTO-EO almost linearly increases with the increasing number of vehicles. Vehicles in MAB require only one iteration information exchange to get RSUs' resource status, so its delay is only 4 ms. The decision making time of both the FDTO and MAB algorithms almost remains static with the increasing number of vehicles, so it is robust to the network size. For example, when there are 10,000 vehicles in the system, the decision making time of CTO-EO, CTO-DO, NGTO, FDTO and MAB is 112,397 ms, 104,691 ms, 36,759 ms, 31 ms and 4 ms, respectively, which indicates that CTO and NGTO cannot provide real-time offloading decisions for highly dynamic and large-scale edge computing systems.

Fig. 16 shows that the average energy consumption increases almost linearly with the increase of the task arrival rate. At the same task arrival rate, the average energy consumption of other algorithms is almost the same except CTO-EO. The energy consumption of our FDTO algorithm is 24.9% higher than that of CTO-EO. For example, when the task arrival rate is 1 tasks/s, the average energy consumptions of CTO-DO, NGTO, FDTO, MAB and CTO-EO are 59.9 W, 59.5 W, 59.1 W, 57.2 W and 47.3 W, respectively.

3) Impact of Mobility and Dynamics on Offloading: We simulate a scenario where 10,000 vehicles are driving through the simulated city area, and they pick a random direction when they come to an intersection. The task generation of each vehicle follows a Poisson process with arrival rate 1 task/s. In each time slot, the system updates the offloading decisions for each vehicle regularly. If a vehicle leaves the transmission range of an RSU or failure connection occurs in a time slot, it will run the vehicle side algorithm to obtain a new offloading decision, rather than waiting till the end of the current slot.

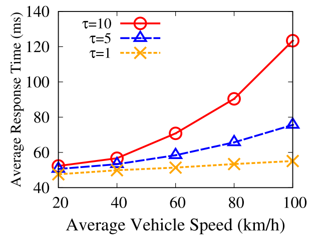


Fig. 17. Average response time vs. average vehicle speed.

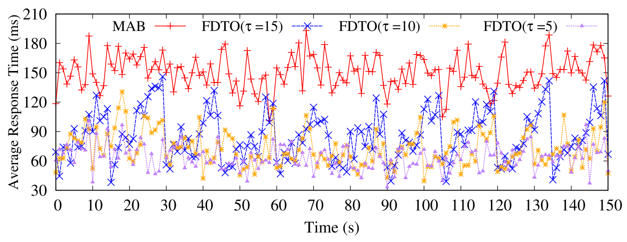


Fig. 18. Average response time vs. time.

Fig. 17 shows the average response time when the average speed of vehicles increases from 20 km/h to 100 km/h given time slots of 1 s, 5 s and 10 s, respectively. In this figure, when the vehicles' average speed is low, it still keeps effective even with a relatively long time slot (e.g., 10 s). For example, when the average speed is 20 km/h, the average response time is 47.6 ms, 50.6 ms and 52.3 ms for time slots of 1 s, 5 s and 10 s, respectively. However, when the vehicle speed is high, the short time slot is needed so as to achieve good performance.

Fig. 18 shows the dynamic change of average response time of both MAB and FDTO algorithms when time slot τ is set as 5 s, 10 s or 15 s, respectively. Each vehicle travels at speeds randomly drawn from 40–80 km/h. We observe the average response time of tasks over time for 150 s. As shown in Fig. 18, the average response time of FDTO is around 35–150 ms, while that of MAB is 100-190 ms. Moreover, when τ is set as 5 s, FDTO can maintain the average response time at 35–90 ms. When τ is increased to 15 s, the average response time of FDTO increases to 35–150 ms. The simulation results indicate that the FDTO algorithm can not only provide real-time decision for highly dynamic and large-scale edge computing scenarios, but also reduce the average response time by 50%–65% compared with the MAB algorithm.

4) Summary of Evaluation Results: From the simulation results in Figs. 5–18 and Table IV, we can make the following three conclusions. First, by Figs. 5–11 and Table IV, our FDTO can control decision making time less than 50 ms by setting the proper values of parameters. Second, by Figs. 12–16, our algorithm can not only provide real-time offloading decisions, but also achieve promising average response time compared with the optimal. Third, by Figs. 17 and 18, our algorithm with adjustable time slot duration can better deal with different mobile scenarios and reduce the task average response time by 50%-65% compared with the existing solutions even in highly dynamic scenarios.

VI. Conclusion

In this paper, we have investigated distributed decision making schemes for task offloading in VEC, which is a promising paradigm for future transportation systems. We have considered the unique challenges of vehicular networks, such as resource constraints, high mobility and large scale, and proposed a fully distributed task offloading (FDTO) scheme that allows vehicles to make real-time offloading decisions only based on incomplete information from neighboring RSUs. We prove that our proposed algorithms will converge to the global optimum by iterations. Extensive simulation results have demonstrated that the proposed solutions can achieve near-optimal task response time with a short time for making offloading decisions.

References

[1] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017.

[2] O. S. Oubbati, M. Atiquzzaman, P. Lorenz, A. Baz, and H. Alhakami, "SEARCH: An SDN-enabled approach for vehicle path-planning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14523–14536, Dec. 2020.

[3] A. Nazemi, Z. Azimifar, M. J. Shafiee, and A. Wong, "Real-time vehicle make and model recognition using unsupervised feature learning," *IEEE Trans. Intell. Transp. Sys.*, vol. 21, no. 7, pp. 3080–3090, 2019.

[4] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[5] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surv. Tuts.*, vol. 19, no. 4, pp. 2322–2358, Fourth Quarter 2017.

[6] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[7] H. Wang, H. Xu, H. Huang, M. Chen, and S. Chen, "Robust task offloading in dynamic edge computing," *IEEE Trans. Mobile Comput.*, vol. 22, no. 1, pp. 500–514, Jan. 2023.

[8] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[9] R. Xie, Q. Tang, Q. Wang, X. Liu, F. R. Yu, and T. Huang, "Collaborative vehicular edge computing networks: Architecture design and research challenges," *IEEE Access*, vol. 7, pp. 178942–178952, 2019.

[10] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint offloading and resource allocation in vehicular edge computing and networks," in *Proc. IEEE Glob. Commun. Conf.*, 2018, pp. 1–7.

[11] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[12] B. Yang, X. Cao, J. Bassey, X. Li, and L. Qian, "Computation offloading in multi-access edge computing: A multi-task learning approach," *IEEE Trans. Mobile Comput.*, vol. 20, no. 9, pp. 2745–2762, Sep. 2021.

[13] K. Xiong, S. Leng, C. Huang, C. Yuen, and Y. L. Guan, "Intelligent task offloading for heterogeneous V2X communications," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2226–2238, Apr. 2021.

[14] Y. Wang et al., "A game-based computation offloading method in vehicular multiaccess edge computing networks," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 4987–4996, Jun. 2020.

[15] H. Wang, T. Lv, Z. Lin, and J. Zeng, "Energy-delay minimization of task migration based on game theory in MEC-assisted vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8175–8188, Aug. 2022.

[16] M. D. Hossain et al., "Dynamic task offloading for cloud-assisted vehicular edge computing networks: A non-cooperative game theoretic approach," *Sensors*, vol. 22, no. 10, 2022, Art. no. 3678.

[17] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for Internet of Vehicles: Model and algorithm," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 5913–5925, Sep. 2021.

[18] Y. Sun, X. Guo, S. Zhou, Z. Jiang, X. Liu, and Z. Niu, "Learning-based task offloading for vehicular cloud computing systems," in *Proc. IEEE Int. Conf. Commun.*, 2018, pp. 1–7.

[19] Z. Zhou, H. Liao, X. Zhao, B. Ai, and M. Guizani, "Reliable task offloading for vehicular fog computing under information asymmetry and information uncertainty," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 8322–8335, Sep. 2019.

[20] K. Xiong, Z. Wang, S. Leng, and J. He, "A digital twin empowered lightweight model sharing scheme for multi-robot systems," *IEEE Internet Things J.*, vol. 10, no. 19, pp. 17231–17242, Oct. 2023.

[21] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2651–2664, Dec. 2018.

[22] K. Zhang et al., "Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks," *IEEE Access*, vol. 4, pp. 5896–5907, 2016.

[23] 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.

[24] E. Altman, U. Ayesta, and B. J. Prabhu, "Load balancing in processor sharing systems," *Telecommun. Syst.*, vol. 47, no. 1/2, pp. 35–48, 2011.

[25] J. Walrand, *An Introduction to Queueing Networks*. Englewood Cliffs, NJ, USA: Prentice Hall, 1988.

[26] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for mobile edge computing in dense networks," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, 2018, pp. 207–215.

[27] L. Chen, S. Zhou, and J. Xu, "Computation peer offloading for energy-constrained mobile edge computing in small-cell networks," *IEEE/ACM Trans. Netw.*, vol. 26, no. 4, pp. 1619–1632, Aug. 2018.

[28] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," 2016, *J. Mach. Learn. Res.*, vol. 17, pp. 2909–2913, 2016.

[29] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter, "Branching and bounds tightening techniques for non-convex MINLP," *Optim. Methods Softw.*, vol. 24, no. 4/5, pp. 597–634, 2009.

[30] X. Cao et al., "Reconfigurable intelligent surface-assisted aerial-terrestrial communications via multi-task learning," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 10, pp. 3035–3050, Oct. 2021.

[31] T. Kumrai, K. Ota, M. Dong, and P. Champrasert, "RSU placement optimization in vehicular participatory sensing networks," in *Proc. IEEE Conf. Comput. Commun. Workshops*, 2014, pp. 207–208.

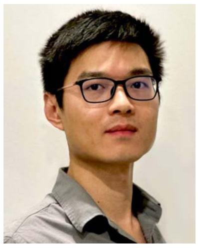
[32] D. Jiang and L. Delgrossi, "IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments," in *Proc. VTC Spring-IEEE Veh. Technol. Conf.*, 2008, pp. 2036–2040.

[33] A. Paier et al., "Average downstream performance of measured IEEE 802.11 p infrastructure-to-vehicle links," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2010, pp. 1–5.

[34] T. S. Rappaport, *Wireless Communications: Principles and Practice, 2/E*. Englewood Cliffs, NJ, USA: Prentice Hall, 2002.

Qianpiao Ma received received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2014 and 2022, respectively. He is currently a Postdoctoral Researcher with Purple Mountain Laboratories. His primary research interests include federated learning, mobile edge computing, and distributed machine learning.

Hongli Xu (Member, IEEE) received the Ph.D. degree in computer software and theory from the University of Science and Technology of China, Hefei, China, in 2007. He is currently a Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored more than 100 papers in famous journals and conferences. His primary research interests include software defined networks, edge computing, and Internet of Thing. He was the recipient of the Best Paper Awards in several famous conferences.



Haibo Wang (Member, IEEE) received the B.E. and Master's degrees in computer science from the University of Science and Technology of China, Hefei, China, in 2016 and 2019, respectively, and the Ph.D. degree in computer science from the University of Florida, Gainesville, FL, USA. He is currently an Assistant Professor with the Department of Computer Science, University of Kentucky, Lexington, KY, USA. His main research interests include Internet traffic measurement, Big Data analytics, software defined networks, and Internet of Things. His was the recipient of IEEE ICNP 2021 Best Paper Award.

Yang Xu (Member, IEEE) received the B.S. degree from the Wuhan University of Technology, Wuhan, China, in 2014, and the Ph.D. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2019. He is currently an Associate Researcher with the School of Computer Science and Technology, University of Science and Technology of China. His primary research interests include ubiquitous computing, deep learning, and mobile edge computing. His work was the recipient of ACM UbiComp 2016 Best Paper Award.

Qingmin Jia received the B.S. degree in communication engineering from Qingdao University of Technology, Qingdao, China, in 2014, and the Ph.D. degree in information and communication engineering from Beijing University of Posts and Telecommunications, Beijing, China, in 2019. He is currently a Researcher with Purple Mountain Laboratories, Nanjing, China. His research interests include edge intelligence, computing and network convergence, and industrial internet of Things.

Chunming Qiao (Fellow, IEEE) is currently a SUNY Distinguished Professor and also the Chair with the Computer Science and Engineering Department, University at Buffalo, Amherst, NY, USA. His current focus is on connected and autonomous vehicles. He has authored extensively with an h-index of over 69. Two of his papers was the recipient of the Best Paper Awards from IEEE and joint ACM/IEEE venues. He was an elected IEEE Fellow for his contributions to optical and wireless network architectures and protocols.