

# BARGAIN-MATCH: A Game Theoretical Approach for Resource Allocation and Task Offloading in Vehicular Edge Computing Networks

Zemin Sun, *Student Member, IEEE*, Geng Sun<sup>✉</sup>, *Member, IEEE*, Yanheng Liu<sup>✉</sup>, Jian Wang<sup>✉</sup>, *Member, IEEE*, and Dongpu Cao, *Senior Member, IEEE*

**Abstract**—Vehicular edge computing (VEC) is emerging as a promising architecture of vehicular networks (VNs) by deploying the cloud computing resources at the edge of the VNs. However, efficient resource management and task offloading in the VEC network is challenging. In this work, we first present a hierarchical framework that coordinates the heterogeneity among tasks and servers to improve the resource utilization for servers and service satisfaction for vehicles. Moreover, we formulate a joint resource allocation and task offloading problem (JRATOP), aiming to jointly optimize the intra-VEC server resource allocation and inter-VEC server load-balanced offloading by stimulating the horizontal and vertical collaboration among vehicles, VEC servers, and cloud server. Since the formulated JRATOP is NP-hard, we propose a co-operative resource allocation and task offloading algorithm named BARGAIN-MATCH, which consists of a bargaining-based incentive approach for intra-server resource allocation and a matching method-based horizontal-vertical collaboration approach for inter-server task offloading. Besides, BARGAIN-MATCH is proved to be stable, weak Pareto optimal, and polynomial complex. Simulation results demonstrate that the proposed approach achieves superior system utility and efficiency compared to the other methods, especially when the system workload is heavy.

**Index Terms**—Game theory, resource allocation, task offloading, vehicular edge computing, vehicular network.

## I. INTRODUCTION

WITH the development of vehicular networks (VNs) and the ever-increasing number of vehicles on the road, various and explosive applications are emerging such as

Manuscript received 7 October 2022; revised 14 January 2023; accepted 19 January 2023. Date of publication 23 January 2023; date of current version 8 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 62172186, 62002133, 61872158, and 62272194, in part by the Science and Technology Development Plan Project of Jilin Province under Grants 20210101183JC and 20210201072GX, and in part by the Young Science and Technology Talent Lift Project of Jilin Province under Grant QT202013. Recommended for acceptance by D. Niyato. (*Corresponding author: Geng Sun.*)

Zemin Sun, Geng Sun, Yanheng Liu, and Jian Wang are with the College of Computer Science and Technology, Jilin University, Changchun 130012, China, and also with the Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China (e-mail: sunzemin@jlu.edu.cn; sungeng@jlu.edu.cn; yhlui@jlu.edu.cn; wangjian591@jlu.edu.cn).

Dongpu Cao is with the School of Vehicle and Mobility, Tsinghua University, Beijing 100190, China (e-mail: dongpu.cao@ieee.org).

This article has supplementary downloadable material available at <https://doi.org/10.1109/TMC.2023.3239339>, provided by the authors.

Digital Object Identifier 10.1109/TMC.2023.3239339

autonomous driving, auto navigation, and augmented reality. These vehicular applications usually require extensive computation resources and low or ultra-low latency. However, fulfilling the computation-intensive and delay-sensitive tasks is challenging due to the limited computation resources of vehicles. To overcome this challenge, mobile edge computing or multi-access edge computing (MEC) [1] is emerging as a promising technology by shifting the cloud computing resources in close proximity to mobile terminals, leading to the new paradigm of vehicular edge computing (VEC) [1], [2], [3]. The VEC migrates the lightweight and ubiquitous resources from cloud servers to the road side units (RSUs) equipped with VEC servers to extend the computation capabilities of the conventional VNs [4]. By offloading the tasks to the VEC servers, the communication latency between the vehicles and the cloud server can be reduced, and the computation overloads on vehicles can be relieved.

However, compared to cloud computing and wireless networks, the VEC network is characterized by the typical features of both edge computing and VNs, i.e., the limited resources of edge servers and highly dynamic of VNs. Therefore, the joint optimization of resource management and task offloading for VEC network is essential, which however confronts several challenges.

*First*, various tasks of vehicles generally arrive dynamically and have stringent requirements on the offloading service. Internally, for a certain VEC server, the limited computational resource of the VEC server and the stringent requirements of requesting vehicles could result in resource competition inside the VEC server, especially in peak hours. Therefore, it is challenging for a VEC server to decide efficient resource allocation policy to satisfy the heterogeneous and stringent requirements of different tasks under the resource constraint. *Second*, high mobility of vehicles and random generation of tasks lead to the spatio-temporally uneven distribution of tasks among VEC servers. Externally, the heterogeneous computational capacity among VEC servers or between VEC server and cloud server further incurs load imbalance and resource under-utilization among the servers. For example, some VEC servers could be overloaded or congested while the others could be underloaded or idle when tasks are randomly offloaded. Therefore, the spatiotemporal heterogeneity among tasks and the computational heterogeneity among servers pose a significant challenge

in designing the efficient task offloading scheme to provide service satisfaction for vehicles and load balance among servers. *Third*, the unique features of VNs, such as the dynamic of channel and mobility of vehicles, add complexity to integrating these features into the optimization for the VEC network.

This work presents a cooperative resource allocation and task offloading approach for VEC network to optimize the resource allocation for servers and offloading satisfaction for vehicles. The main contributions are summarized as follows:

- We employ a hierarchical architecture of resource allocation and task offloading for VEC network to coordinate both the space-time-requirement heterogeneity among tasks and the computational heterogeneity among servers. Specifically, the regional software-defined networking (SDN) [5], which separates the data and control planes, is integrated for efficient resource allocation and task offloading. Under the coordination of the controller, the intra-server resource allocation and inter-server load-balanced offloading are optimized by stimulating horizontal and vertical collaborations among vehicle, edge, and cloud layers.
- We formulate a joint resource allocation and task offloading problem (JRATOP) by jointly optimizing the strategies of resource allocation and pricing, and task offloading, with the aim of maximizing the system utility that is theoretically modeled by synthesizing the unique features of VN channels, nonorthogonal multiple access (NOMA) [6], mobility of vehicles, spatiotemporal heterogeneity of tasks, computational heterogeneity of servers, and energy consumption of nodes.
- Due to the NP-hardness of JRATOP, we propose a cooperative resource allocation and task offloading algorithm BARGAIN-MATCH that consists of two components. For intra-server resource allocation, a bargaining game-based incentive approach is proposed to stimulate collaboration between the task (of a vehicle) and a VEC/cloud server for resource allocation and pricing. For inter-server task offloading, a many-to-one matching is constructed between tasks and servers to stimulate edge-edge collaboration for horizontal task offloading and edge-cloud collaboration for vertical task offloading.
- The performance of BARGAIN-MATCH is verified through theoretical analysis and simulation. Specifically, BARGAIN-MATCH is proved to be stable, weak Pareto optimal, and polynomial complex. Furthermore, simulation results show that BARGAIN-MATCH can achieve superior system utility and system efficiency, especially when the workload is heavy.

The remaining of this paper is organized as follows. Section II reviews the related work. Section III presents the models and preliminaries. The problem formulation is given in Section IV. Section V elaborates the proposed BARGAIN-MATCH. Section VI shows the simulation results and discussions. In Section VII-A, we extend the investigation to the scenario with the real-world vehicle applications. This work is concluded in Section VIII. Furthermore, for the sake of readability, all notations are listed in Table I.

## II. RELATED WORK

In recent years, MEC has emerged as a promising paradigm to provide cloud-computing capabilities at the network edges by deploying lightweight MEC servers ubiquitously in close proximity to end users [1], [7]. Delay-sensitive, computation-intensive, and energy-consuming computation tasks can be offloaded to the MEC servers to improve the quality of service for mobile users. The task offloading and/or resource management for MEC networks has attracted increasing research attempts. Several studies [8], [9], [10] focus on the joint task offloading and resource allocation in the single (or double)-user and single-server MEC system, which may not be applicable to the real-world scenarios, especially those with heavy or bursty workloads.

To overcome the above challenge, more studies focus on the multi-user and multi-server MEC environment regarding the task offloading and resource allocation. For example, in [11], a new scheme is proposed to guarantee the efficiency and reliability of mission-critical task offloading and resource allocation by imposing probabilistic and statistical constraints to the task queue length based on extreme value theory. Apostolopoulos et al. [12] present a novel risk-aware data offloading approach where the risk-seeking offloading behavior of users and the resource over-exploitation of MEC servers are jointly considered by using the prospect theory and tragedy of the commons. Zhang et al. [13] employs Lyapunov optimization theory to optimize the task offloading and resource allocation in the MEC-based cloud radio access network, aiming at maximizing the network energy efficiency. Tan et al. [14] construct a two-level framework for energy-efficient task offloading and resource allocation in OFDMA-based MEC networks. However, these studies mainly focus on MEC networks, which may be not applicable for VNs with highly dynamic vehicles and wireless channels, and uneven distributed nodes and workloads.

In recent years, considerable efforts have been made to improve the performance of VEC network, especially focusing on task offloading to mitigate the competition among vehicles [15], [16], to balance the workload [4], [17], and to explore the available resources of vehicles [18], [19], [20]. Sun et al. [15] propose an adaptive learning based task offloading algorithm based on the multi-armed bandit theory to minimize the average offloading delay. Wang et al. [16] propose a multi-user non-cooperative computation offloading game, where each vehicle decides whether to offload its task to the VEC server according to the traffic density. An SDN-based VEC architecture is introduced in [17] to provide centralized network management to balance the workload of task offloading. Dai et al. [4] construct a cooperative task offloading mechanism based on the queuing theory to minimize the task completion delay and balance the workload at edges. Qin et al. [18] focus on exploiting vehicles' idle and redundant resources for energy efficient task offloading in VNs under information uncertainty. Liu et al. [19] propose a task offloading scheme by exploiting multi-hop vehicle computation resources in VEC networks. Wang et al. [20] consider the available neighboring VEC clusters and propose an imitation learning-based task scheduling approach to minimize

TABLE I  
SUMMARY OF NOTATIONS

Symbol	Description	Symbol	Description
$a \in \{0, \mathcal{E}, o\}$	The offloading destination of a task	$\alpha_i$	The effective switched capacitance of vehicle $i$ 's/server $j$ 's CPU
$\beta^L / \beta^{NL}$	The path loss exponent for LoS/NLoS communication	$B_{i,j}$	The bandwidth
$c_{j,i}(t) \in [c_{j,i}^{\min}(t), c_{j,i}^{\max}(t)]$	The unit price of resources by server $j$	$c$	The speed of light
$C_i(t)$	The computation resources required by per bit of task	$C_i^{\text{req}}(t)$	The amount of computation resources
$C_i^a(t)/C_j^i(t)$	The normalized cost of vehicle $i$ /server $j$	$C_i^{\max}/C_j^{\max}$	The maximum budget of vehicle $i$ /server $j$
$\Delta c_{j,i}(t)$	Bid-spread ask	$\Delta d_{i,j}(t)$	The horizontal distance difference between the current and previous time epochs
$d_{i,j}(t)$	The distance between $i$ and $j$	$d_0$	The reference distance
$\mathcal{D}_i^{\text{in}}(t)$	The size of the computation task	$\mathcal{D}_i^{\text{out}}(t)$	The size of the computation result
$\delta_i^{i*}(t)/\delta_j^{j*}(t)$	The optimal partitions when vehicle $i$ makes a proposal	$\delta_i^{i*}(t)/\delta_j^{j*}(t)$	The optimal partitions when server $j$ makes a proposal
$D_j$	The set of less preferred tasks of server $j$	$E_i^0(t)/E_j^i(t)$	The energy consumption of vehicle $i$ /server $j$
$E_i^{\max}/E_j^{\max}$	The energy constraint of vehicle $i$ /server $j$	$\epsilon_i(t)/\epsilon_j(t)$	The discount factor of vehicle $i$ /server $j$
$\mathcal{E} = \{1, \dots, E\}$	The set of VEC servers	$f_c$	The carrier frequency
$F(\text{var}_1, c_{j,i}(t))$	The function in Eq. (24)	$f_i(t)$	The available computation capacity of vehicle $i$
$f_i^{\max}/f_j^{\max}$	The computation capability of vehicle $i$ /server $j$	$f_{j,i}(t)/f_{o,i}(t)$	The computation resources provided by VEC server $j$ / cloud server $o$ to vehicle $i$
$g_{i,j}^L(t)/g_{i,j}^{NL}(t)$	The channel power gain between vehicle $i$ and VEC server $j$	$g_{i,j}(t)/g_{I,j}(t)$	The channel power gain between vehicle $i/I$ and VEC server $j$
$h_{i,j}^L(t)/h_{i,j}^{NL}(t)$	The small-scale fading for LoS/NLoS communication	$i \in \mathcal{V}$	The index of vehicle $i$
$j \in \mathcal{E} / j \in \{\mathcal{E}, o\}$	The index of server $j$	$\chi_\sigma^L/\chi_\sigma^{NL}$	The shadowing for LoS/NLoS communication
$L_{i,j}^L(t)/L_{i,j}^{NL}(t)$	The pathloss for LoS/NLoS communication	$m^L/m^{NL}$	The fading parameter for LoS/NLoS communication
$N_i^{\text{core}}/N_j^{\text{core}}$	The number of vehicle $i$ /server $j$ 's CPU cores	$N_0$	The noise power
$\Omega_k^j/\Omega_j^k$	The preference of task $k$ /server $j$ towards server $k$ /task $k$	$o$	Cloud server
$\mathbf{P}_i(t) = (X_i(t), Y_i(t), 0)$	The position of vehicle $i$	$p_i^{\text{gen}}(t)$	The indicator of vehicle $i$ 's task generation
$\mathbf{P}_j = (X_j, Y_j, 0)$	The position of VEC server $j$	$P_i(t)/P_I(t)$	The transmit power of vehicle $i/I$
$p_{i,j}^L$	The probability of LoS transmission between vehicle $i$ and VEC server $j$	$\Phi$	Matching result
$(\mathcal{P}_B, \mathcal{S}_B, \mathcal{U}_B)$	The triplet of bargaining	$(\mathcal{P}_M, \Omega, \Phi)$	The triplet of matching
$r_c$	The data rate between edge and cloud	$r_f$	The data rate of fiber link
$\rho_k(j)/\rho_j(k)$	The preference of task $k$ /server $j$ on server $j$ /task $k$	$\Phi(k)/\Phi(j')$	The matching list of task $k$ /server $j'$
$R_j(j \in \mathcal{E})$	The coverage radius of VEC server $j$	$\mathcal{R}_j^i(t)$	The normalized revenue of server $j$
$r_{i,j}(t)$	The data transmission rate between vehicle $i$ and VEC server $j$	$SW$	Social welfare
$s_i^a(t)$	The offloading strategy of task $\mathcal{T}_i$	$\sigma^L/\sigma^{NL}$	The standard deviation of shadowing for LoS/NLoS transmission
$s_{j,i}^*(t) = (f_{j,i}^*(t), c_{j,i}^*(t))$	The optimal resource allocation and pricing	$S_{\text{off}}^*(t)/S_{\text{all}}^*(t)$	The optimal strategy of offloading/resource allocation
$t \in \mathbf{T} = \{0, \dots, T-1\}$	System timeline	$\Delta t$	Each slot duration
$\mathbf{T}(t_0)$	Time epoch	$T_0$	Each time epoch duration
$T_{i,j}^{\text{soj}}$	The sojourn time of vehicle $i$ in the coverage of VEC server $j$	$\tau > 0$	CPU parameter
$\mathcal{T}_i(t)$	The task generated by vehicle $i$ at time $t$	$T_i^{\max}(t)$	The maximum tolerable delay of task $i$
$T_i^0(t)/T_i^j(t)/T_i^o(t)$	The total delay for offloading task $\mathcal{T}_i(t)$ locally/on edge server $j$ /on cloud server $o$	$T_{i,j}^{\text{tran}}(t)$	The delay for vehicle $i$ to transmit the task to VEC $j^{\text{cur}}$
$T_{i,j}^{\text{comp}}(t)/T_{i,o}^{\text{comp}}(t)$	The computation delay that the task is processed by VEC server $j$ /cloud server $o$	$\tau > 0$	CPU parameter
$T_{j,\text{cur}}^{\text{hand}}(t)/T_{j,\text{cur},o}^{\text{hand}}(t)$	The horizontal/vertical task handover delay	$\mathcal{T}^{\text{rej}}(t)$	The tasks that are rejected
$T_{j,\text{cur}}^{\text{hand}}(t)/T_{o,j,\text{cur}}^{\text{hand}}(t)$	The horizontal/vertical result handover delay	$\Psi_i^a(t)$	The normalized satisfaction level of task completion delay
$\mathcal{V}_j = \{1, \dots, V_j\}$	The set of interference vehicles within the range of VEC server $j$	$U_i^a(t)/U_j^i(t)$	The utility obtained by vehicle $i$ /server $j$ from offloading task $\mathcal{T}_i(t)$
$\mathcal{V} = \{1, \dots, V\}$	The set of vehicles	$\mathcal{V}^{\text{req}}(t)$	The requesting vehicle set
$v_i(t) \in [v^{\min}, v^{\max}]$	The velocity of vehicle $i \in \mathcal{V}$	$w_i/w_j$	The weight coefficient of vehicle $i$ /server $j$
$\zeta_i(t)$	The indicator for the movement direction of vehicle $i$	$P(\zeta_i(t))$	The general model for the movement of vehicles

the system energy consumption. However, these studies mainly focus on optimizing the offloading strategies with insufficient consideration for the resource allocation of VEC servers. Besides, most of them consider the single-server scenario [15], [16], [18], [19], [20].

Considering the limited resources of edge servers, some studies target on resource management for VEC networks, including spectrum sharing [21], computation resource allocation [22], multi-dimensional resource management [23], and the exploitation of under-utilized vehicular resources [20], [24]. For example, Peng et al. [23] employ a deep learning approach to manage the resources of VEC servers for the delay-sensitive applications of vehicles. Zhu et al. [22] adopt a Stackelberg game to model the interaction between vehicles and VEC servers to obtain the price and amount of computation resources to be allocated. In [24], the authors focus on the power-aware resource management to jointly optimize the resource utilization and energy efficiency of VEC servers. However, these studies do not consider the offloading strategies from the perspective of vehicles.

Since the problems of resource allocation and task offloading are coupled with each other, several research efforts have been devoted to joint resource allocation and task offloading, aiming at delay-driven system utility optimization [4], [25], [26], [27], spectrum efficiency improvement [28], or energy efficiency improvement [29], [30]. For instance, Dai et al. [4] propose a task offloading and resource allocation scheme for VEC networks to minimize the task processing delay under the permissible latency constraint. In [25], the task offloading is optimized by maximizing the task completion probability, and the resource allocation is determined by performing a mobility-aware greedy algorithm. Zhou et al. [26] propose an incentive mechanism based on contract-match mechanism to leverage the under-utilized computation resources for task offloading of nearby vehicles. In [27], the optimal offloading decision and the resource allocation are achieved by using the game theory. Li et al. [28] consider the influence of time-varying channel on the time-varying spectrum efficiency of task offloading, which is solved by using the branch and bound algorithm. Considering the limited energy of vehicles, Huang et al. [29], [30] propose an energy efficiency-driven approach to reduce the energy costs of vehicles under the constraint of computation resources of VEC servers.

Although the above-mentioned work has significantly improve the performance of the VEC network, there are still some problems to be addressed. The heterogeneity among tasks and servers, the load imbalance among servers, and the unique features of VN such as the mobility of vehicles, the dynamic of vehicular channel, and the energy consumption of task execution have not been jointly explored for task offloading and resource allocation. Distinguished from the previous works, this work studies cooperative resource allocation and task offloading in the VEC network, where the stringent requirements of tasks, the heterogeneity among tasks and servers, the load imbalance, the unique features of VNs, and the energy limitation of the VEC networks, are jointly considered.

### III. MODELS AND PRELIMINARIES

In this section, a VEC-enabled VN architecture is first introduced, followed by the traffic and mobility model, communication model, computation model, and energy consumption model.

#### A. System Model

1) *System Overview*: As shown in Fig. 1, in the spatial domain, we consider an SDN-based hierarchical architecture for VEC network that consists of a vehicle layer with  $V$  vehicles  $\mathcal{V} = \{1, \dots, i, \dots, V\}$ , an edge layer with  $E$  VEC servers  $\mathcal{E} = \{1, \dots, j, \dots, E\}$  that are mounted on RSUs,<sup>1</sup> a control layer with a regional SDN controller, and a remote cloud layer with a cloud server  $o$ . For simplicity but without loss of generality, we assume that the vehicles and VEC-mounted RSUs in the system model are equipped with single antenna [31], [32]. *At the vehicle layer*, vehicles distribute randomly on the multi-lane bi-directional road, run in either direction, and could generate tasks at any time. Each vehicle independently decides whether to process the task locally or upload it to the connected VEC server<sup>2</sup> by using NOMA. *At the edge layer*, VEC servers are mounted on RSUs that are deployed alongside the road with non-overlapping coverage radius  $R_j, j \in \mathcal{E}$ , and the service area of each VEC server is defined as the wireless coverage of the RSU. Moreover, the VEC servers are connected to the regional SDN controller [33], [34] and to each other via high-speed fiber links [13], [20], [35] and mobile access gateways (MAGs) [36]. Besides, each VEC server is responsible to collect the local information on its own states, vehicles' states, and the channel state information (CSI), and uploads the information to the control layer. *At the control layer*, the regional SDN controller, on which our algorithm runs, is responsible for decision making and handover management with the support of MAG. For decision making, the following cooperative decisions are performed by our algorithm under the coordination of the controller, i.e., intra-VEC server resource allocation and inter-server task offloading that includes the horizontal task migration among servers and vertical task migration from the VEC server and the cloud server. For handover management, it is incorporated into the task migration to guarantee the connectivity for moving vehicles [36], [37]. With the SDN technology, the controller only needs the knowledge acquired from the edge layer, and makes accordingly decisions by performing the proposed algorithm. *At the cloud layer*, the tasks can be vertically offloaded from the edge layer to the cloud layer under the guidance of the controller.

In the temporal domain, the VEC system operates in a discrete time-slotted manner. Specifically, the system timeline is equally divided into  $T$  time slots, i.e.,  $t \in \mathbf{T} = \{0, \dots, T-1\}$ , where each slot duration  $\Delta t$  is consistent with the coherence block of the wireless channel [38]. Every  $T_0$  consecutive slots are grouped into a time epoch  $t_0 \in \mathbb{Z}^+$ , where  $\mathbf{T}(t_0) = \{t_0 -$

<sup>1</sup>In this work, the RSU and VEC server are used interchangeably unless otherwise indicated.

<sup>2</sup>A vehicle directly uploads the tasks to the VEC server within the current service area since it is unaware of the servers out of the service area [20].

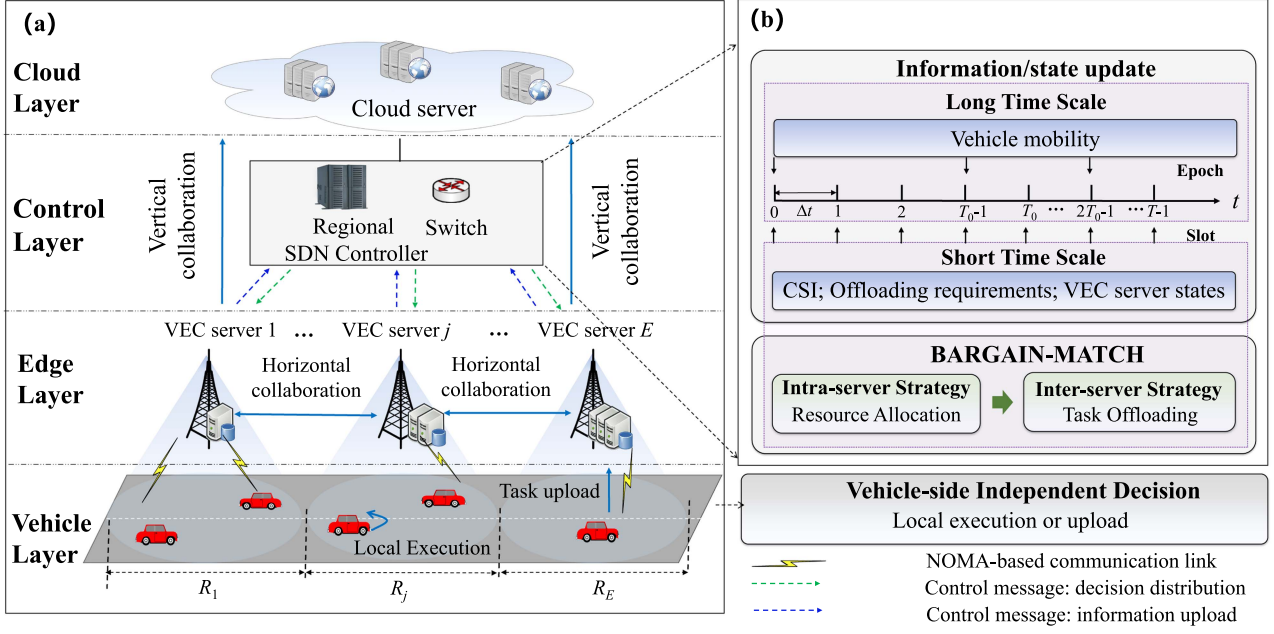


Fig. 1. The cooperative architecture of resource allocation and task offloading in VEC network.

1)  $T_0, \dots, t_0 \cdot T_0 - 1$ . Moreover, the CSI and states of VEC servers and vehicles are captured and updated in different time scales. Specifically, in the short timescale, the information on CSI, offloading requirements of vehicles, and states of VEC servers are captured and updated. In the long timescale, the mobility of vehicles is captured and updated.<sup>3</sup> This is because CSI keeps constant in a time slot and varies across different time slots, while the mobility of a vehicle keeps approximately constant during a few consecutive slots [38].

2) *Basic Models*: The basic models for the nodes in the system is illustrated in the following.

*Mobility Model*. The mobility of vehicle  $i \in \mathcal{V}$  is denoted as  $(\mathbf{P}_i(t), v_i(t))$ , where  $\mathbf{P}_i(t) = (X_i(t), Y_i(t), 0)$  and  $v_i^{\min} \leq v_i(t) \leq v_i^{\max}$  denote the position and velocity of vehicle  $i$  at time  $t$ , respectively. Suppose that vehicle  $i$  is located in the service range of VEC server  $j$  at time  $t$ , the moving direction of vehicle  $i$  is denoted as  $\zeta_i(t)$  is the indicator function of vehicle  $i$ 's movement direction towards VEC server  $j$ .  $\zeta_i(t)$  can be estimated based on the difference of the horizontal distance (between  $i$  and  $j$ ) between the current time epoch and the previous time epoch, i.e.,  $\Delta d_{i,j}(t) = |X_i(\lfloor t/T_0 \rfloor) - X_j| - |X_i(\lfloor t/T_0 \rfloor - 1) - X_j|$ . According to [39], we consider the following two cases. *i*) If the previous location of the vehicle is known,  $\zeta_i(t)$  can be given as an integral indicator, i.e.,  $\zeta_i(t) = \begin{cases} 1, & \Delta d_{i,j}(t) > 0 \\ -1, & \Delta d_{i,j}(t) < 0 \end{cases}$  to indicate the vehicle is moving toward ( $\zeta_i(t) = 1$ ) or moving away from ( $\zeta_i(t) = -1$ ) VEC server  $j$ . *ii*) If vehicle's mobility is not fully known and should be predicted for the future time, i.e.,  $t = 0$  or  $\Delta d_{i,j}(t) = 0$ ,  $\zeta_i(t) = \pm P(\zeta_i(t))$  is defined as a continuous variable in  $[-1, 1]$

<sup>3</sup>The position and velocity of vehicles can be estimated from the GPS data that is known to the RSUs [26].

to indicate the probability of vehicle's movement direction, where  $P(\zeta_i(t))$  is a general model to represent the movement of vehicles which can be set as a typical mobility model, e.g., Markovian mobility [40]. Therefore,  $\zeta_i(t)$  can be concluded as:

$$\zeta_i(t) = \begin{cases} 1, & t > 0, \Delta d_{i,j^{\text{cur}}}(t) > 0, \\ -1, & t > 0, \Delta d_{i,j^{\text{cur}}}(t) < 0, \\ \pm P(\zeta_i(t)), & t = 0 \text{ or } \Delta d_{i,j^{\text{cur}}}(t) = 0. \end{cases} \quad (1)$$

Given that vehicle  $i$  is located within the coverage of VEC server  $j^{\text{cur}}$  currently, the remaining sojourn time of vehicle  $i$  before leaving the coverage of VEC server  $j$  can be obtained as:

$$T_{i,j}^{\text{soj}} = \frac{R_j + \zeta_i(t) \cdot |X_i(t) - X_j^{\text{cur}}|}{v_i(t)}. \quad (2)$$

where  $R_j$  is the communication radius of VEC server  $j$ ,  $|X_i(t) - X_j|$  is the horizontal distance between vehicle  $i$  and VEC server  $j$ .

Furthermore, the VEC server  $j^{\text{arr}}$  that vehicle  $i$  will be attached after time  $T_i^{\text{move}}$  can be estimated as:

$$j^{\text{arr}} = j + \zeta_i(t) \cdot \left\lceil \frac{v_i(t) \cdot T_i^{\text{move}} - (R_j + \zeta_i(t) \cdot |X_i(t) - X_j|)}{v_i(t)} \right\rceil. \quad (3)$$

*Vehicle Model*. Each vehicle  $i \in \mathcal{V}$  is characterized by  $(\mathbf{P}_i(t), v_i(t), f_i^{\text{max}}, p_i^{\text{gen}}(t))$ , where  $\mathbf{P}_i(t) = (X_i(t), Y_i(t), 0)$  and  $v_i(t)$  denote the position and the velocity of vehicle  $i$  at time  $t$ , respectively, and  $f_i^{\text{max}}$  is the computation capability of vehicle  $i$ . Moreover, each vehicle can generate multiple tasks during the system timeline, where in each time slot the vehicle could generate one task or not. The task generation of vehicle  $i$  in time slot  $t$  is denoted by a binary indicator  $p_i^{\text{gen}}(t) \in \{0, 1\}$ , where  $p_i^{\text{gen}}(t) = 1$  means that vehicle  $i$  generates a task. Considering the resource limitation, each vehicle is assumed to be equipped with one CPU core [4].

**Task Model.** The task generated by vehicle  $i$  in time slot  $t$  is denoted as  $\mathcal{T}_i(t) = (\mathcal{D}_i^{\text{in}}(t), \mathcal{D}_i^{\text{out}}(t), \mathcal{C}_i(t), \mathcal{C}_i^{\text{req}}(t), T_i^{\text{max}}(t))$ , where  $\mathcal{D}_i^{\text{in}}(t)$  is the size of the input computation task,  $\mathcal{D}_i^{\text{out}}(t)$  is the size of the computation result,  $\mathcal{C}_i(t)$  is the computation resources (CPU cycles/s) required by per bit of task (i.e., the computation intensity),  $\mathcal{C}_i^{\text{req}}(t) = \mathcal{D}_i^{\text{in}}(t) \cdot \mathcal{C}_i(t)$  is the amount of computation resources that is required to fulfill the task, and  $T_i^{\text{max}}$  is the maximum tolerable delay of the task.

**Server Model.** First, the local of each edge server  $j \in \{\mathcal{E}\}$  is denoted as  $\mathbf{P}_j = (X_j, Y_j, 0)$ . Furthermore, the heterogeneous computational capabilities of edge and cloud servers are characterized by different numbers of CPU cores and different amounts of computation resources of each CPU core, i.e.,  $j \in \{\mathcal{E}, o\}$  is characterized by  $(f_j^{\text{max}}, N_j^{\text{core}})$ , where  $N_j^{\text{core}}$  denotes the number of CPU cores of server  $j$  and  $f_j^{\text{max}}$  (cycles/s) denotes the maximal computational resource of server  $j$ . We consider that the servers are equipped with multi-core CPUs [4], [41] so that multiple tasks can be processed by a server in parallel, and each CPU core is dedicated to at most one task in each time slot [41].

**Strategy Variables.** The following strategies are determined jointly. For task  $\mathcal{T}_i(t)$ , the offloading strategy is defined as a binary variant  $s_i^a(t) \in \{0, 1\}$ ,  $a \in \{0, \mathcal{E}, o\}$ , where  $a$  denotes the offloading destination of the task. Specifically, the task  $\mathcal{T}_i(t)$  could be executed on vehicle  $i$  ( $s_i^0(t) = 1$ ), on VEC server  $j \in \mathcal{E}$  ( $s_i^j(t) = 1$ ), or on cloud server  $o$  ( $s_i^o(t) = 1$ ). For server  $j \in \{\mathcal{E}, o\}$ , the resource allocation strategy is denoted as  $\{(f_{j,i}(t), c_{j,i}(t))\}$ , where  $f_{j,i}(t)$  is the amount of computation resources provided by server  $j$  to vehicle  $i$  at time  $t$ , and  $c_{j,i}(t)$  is the unit price of computation resources charged to vehicle  $i$  for executing task  $\mathcal{T}_i(t)$ .

## B. Communication Model

This work mainly focuses on the task uploading from vehicles to edge servers, and the task downloading from edge servers to vehicles is omitted since the size of the computation outcome is generally much smaller than that of the computation input for most mobile applications [42], [43], [44]. For task uploading, the power-domain NOMA is employed to support multiple requesting vehicles to simultaneously transmit their tasks to the RSU with which they are currently attached, considering the following reasons. In terms of the advantage of the power-domain NOMA, it has been recognized as a promising technique to achieve the capacity region for multi-user uplink communications in MEC networks [45], [46], [47], vehicular networks [48], unmanned aerial vehicle (UAV) networks [49], etc. Specifically, in the power-domain NOMA, multiple vehicles are able to share the same bandwidth resources, and they are distinguished in the power domain with the aid of the key technique of successive interference cancellation (SIC) [50], [51]. In terms of the practicality of task uploading, when performing SIC, the SIC receiver decodes the stronger signals sequentially from the superimposed signals by treating the weaker signals as noise, implying that the performance of the power-domain NOMA is directly influenced by the capability of the SIC receivers [31]. Specifically, in the power-domain NOMA system, the SIC implementation has a linear computational complexity in the number of users [52].

For the downlink scenario, the linear complexity may be a bottleneck since it requires the implementation of a sophisticated SIC scheme at each receiver with limited processing capabilities. However, for the uplink scenario that is mainly considered in this work, it is relatively more convenient and affordable for the RSU equipped with more powerful VEC servers when the capability of SIC receivers is taken into careful consideration.

In the considered NOMA-based VEC network, the details of employing the SIC for multi-vehicle task uploading is illustrated as follows. First, to capture the capability of the SIC receiver at the RSU side, we consider that each RSU  $j$  is equipped with an  $S_j$ -SIC receiver [31], [53], [54], where  $S_j$  means that the SIC receiver at RSU  $j$  is capable of successively decoding the signals that are simultaneously transmitted by at most  $S_j$  vehicles within its service range.<sup>4</sup> Furthermore, by using SIC, the channel gains of the uploading vehicles  $\mathcal{V}_j(t) = \{1, \dots, V_j(t)\}$  ( $V_j(t) \leq S_j$ ) within the range of VEC server  $j$  are first ordered as  $g_{1,j}(t) \geq \dots \geq g_{i,j}(t) \dots \geq g_{V_j,j}(t)$ ,  $\forall i \in \mathcal{V}_j(t)$  [56]. To decode the signal of vehicle  $i$ , VEC server  $j$  first decodes the stronger signals of vehicles  $I' < i$ , then subtracts them from the superposed signal, and treats the weaker signals of vehicles  $V_j(t) \geq I \geq i + 1$  as interference. Therefore, the data transmission rate from vehicle  $i$  to VEC server  $j$  using NOMA can be calculated as:

$$r_{i,j}(t) = B_{i,j} \cdot \log_2 \left( 1 + \frac{P_i(t) \cdot g_{i,j}(t)}{N_0 + \sum_{I=i+1}^{V_j(t)} P_I(t) \cdot g_{I,j}(t)} \right), \quad (4)$$

where  $B_{i,j}$  is the bandwidth,  $P_i(t)$  is the transmit power of vehicle  $i$ ,  $g_{i,j}(t)$  is the channel power gain between vehicle  $i$  and VEC server  $j$ ,  $N_0$  is the noise power,  $P_I(t)$  is the transmit power of interference vehicle  $I \in \mathcal{V}_j(t)$ , and  $g_{I,j}(t)$  is the channel power gain between the interference vehicle and the VEC server.

The channel power gain is calculated by incorporating the probabilistic LoS and NLoS transmissions into both small-scale and large-scale fading [57]. For uplink communication, the channel power gain between vehicle  $i$  and VEC server  $j$  at time  $t$  can be given as:

$$g_{i,j}(t) = p_{i,j}^L \cdot g_{i,j}^L(t) + (1 - p_{i,j}^L) \cdot g_{i,j}^{NL}(t), \quad (5)$$

where  $p^L$  denotes the probability of LoS transmission and  $g_{i,j}^L(t)/g_{i,j}^{NL}(t)$  denotes the channel power gain between vehicle  $i$  and VEC server  $j$  for LoS / NLoS transmission, which is calculated as:

$$\begin{cases} g_{i,j}^L(t) = |h_{i,j}^L(t)|^2 \cdot (L_{i,j}^L(t))^{-1} \cdot 10^{-\frac{\chi_{\sigma}^L}{10}}, & (6a) \\ g_{i,j}^{NL}(t) = |h_{i,j}^{NL}(t)|^2 \cdot (L_{i,j}^{NL}(t))^{-1} \cdot 10^{-\frac{\chi_{\sigma}^{NL}}{10}}, & (6b) \end{cases}$$

where  $h_{i,j}^L(t)/h_{i,j}^{NL}(t)$ ,  $L_{i,j}^L(t)/L_{i,j}^{NL}(t)$ , and  $\chi_{\sigma}^L/\chi_{\sigma}^{NL}$  denote the components of *small-scale fading*, *pathloss*, and *shadowing*, respectively for LoS/NLoS communication. For LoS communication, these components are detailed as follows. *i) The small-scale fading* characteristic of the channel is captured by using a parametric-scalable and good-fitting generalized fading model, i.e., Nakagami- $m$  fading [58]. Specifically,

<sup>4</sup>The value of  $S_j$  depends on the architecture types of SIC receivers (e.g., hard-based or soft-based SIC) in practical systems, which further relies on the capability of the RSU [55].

$h_{i,j}^L(t)/h_{i,j}^{NL}(t)$  follows a Nakagami distribution with fading parameter  $m^L/m^{NL}$ , i.e.,  $\begin{cases} h_{i,j}^L(t) \sim f_A(h_{i,j}^L(t), m^L) \\ h_{i,j}^{NL}(t) \sim f_A(h_{i,j}^{NL}(t), m^{NL}) \end{cases}$ ,  $f_A(h, m) = \frac{2(m)^m \cdot h^{2m-1} \cdot e^{-\frac{m \cdot h^2}{\bar{p}}}}{\Gamma(m) \cdot \bar{p}^m}$ , where  $0.5 \leq m \leq 5$  denotes the Nakagami fading parameter,  $\bar{p}$  is the average received power in the fading envelope, and  $\Gamma(m)$  is the Gamma function. *ii) The path loss* between vehicle  $i$  and VEC server  $j$  for LoS/NLoS communication can be calculated as  $\begin{cases} L_{i,j}^L(t) = \frac{(4\pi \cdot d_0 \cdot f_c)^2}{c^2} \cdot \left(\frac{d_{i,j}(t)}{d_0}\right)^{\beta^L} \\ L_{i,j}^{NL}(t) = \frac{(4\pi \cdot d_0 \cdot f_c)^2}{c^2} \cdot \left(\frac{d_{i,j}(t)}{d_0}\right)^{\beta^{NL}} \end{cases}$ , where  $f_c$  is the carrier frequency,  $c$  is the speed of light,  $d_0$  is the reference distance,  $d_{i,j}(t)$  is the distance between  $i$  and  $j$ , and  $\beta^L/\beta^{NL}$  is the path loss exponent for LoS/NLoS communication. *iii) The shadowing* captures the signal attenuation caused by shadowing in transmission, which is a zero-mean Gaussian distributed random variable, i.e.,  $\begin{cases} \chi_\sigma^L(t) \sim \mathcal{N}(0, (\sigma^L)^2) \\ \chi_\sigma^{NL}(t) \sim \mathcal{N}(0, (\sigma^{NL})^2) \end{cases}$ , where variance  $\sigma^L/\sigma^{NL}$  denotes the standard deviation of shadowing for LoS/NLoS transmission [57]. Similarly, the channel power gain  $g_{I,i}(t)$  between interference vehicle  $I$  and VEC server  $j$  can be easily obtained by replacing the corresponding parameters.

### C. Offloading Delay and Energy Consumption

*1) Offloading Delay:* For task  $\mathcal{T}_i(t)$  generated by vehicle  $i$  at time slot  $t$ , the service delay of completing the task depends on the offloading strategy  $s_i^a$ .

*Local Offloading.* When task  $\mathcal{T}_i(t)$  is processed by vehicle  $i$  locally, the delay is given as:

$$T_i^0(t) = \frac{\mathcal{C}_i^{\text{req}}(t)}{f_i(t)}, \quad (7)$$

where  $f_i(t)$  is the available computation capacity of vehicle  $i$  at time  $t$ .

*Edge Offloading.* When task  $\mathcal{T}_i(t)$  is processed by VEC server  $j$ , the offloading delay mainly consists of transmission delay, computation delay and horizontal migration delay,<sup>5</sup> i.e.,

$$T_i^j(t) = \underbrace{T_{i,j^{\text{cur}}}^{\text{tran}}(t)}_{\text{Transmission}} + \underbrace{T_{i,j}^{\text{comp}}(t)}_{\text{Computation}} + \underbrace{T_{j,j^{\text{arr}}}^{\text{hand}}(t)\mathbb{I}_{(j \neq j^{\text{cur}})} + T_{j,j^{\text{arr}}}^{\text{hand}}(t)\mathbb{I}_{(j \neq j^{\text{arr}})}}_{\text{Horizontal migration}}. \quad (8)$$

*i)  $T_{i,j^{\text{cur}}}^{\text{tran}}(t) = \frac{\mathcal{D}_i^{\text{in}}(t)}{r_{i,j}(t)}$*  is the delay for vehicle  $i$  to transmit the task to VEC server  $j^{\text{cur}}$  where the vehicle is within the service area. *ii)  $T_{i,j}^{\text{comp}}(t) = \frac{\mathcal{C}_i^{\text{req}}(t)}{f_{j,i}(t)}$*  is the execution delay of the task, where  $f_{j,i}(t)$  is the computation resources that is allocated by VEC server  $j$  to the task. *iii) Similarly to [37], the horizontal migration delay incorporates task handover delay and result handover delay, i.e.,  $T_{j^{\text{cur}},j}^{\text{hand}}(t) = \frac{2\mathcal{D}_i^{\text{in}}(t)}{r_f}$  and  $T_{j,j^{\text{arr}}}^{\text{hand}}(t) = \frac{2\mathcal{D}_i^{\text{out}}(t)}{r_f}$ , where  $r_f$*

is the data rate of fiber link. On the one hand, when task is offloaded to the VEC server in whose coverage the vehicle is currently located, the task handover is used to forward the task first from the source VEC server  $j^{\text{cur}}$  to the controller, and then from the controller to the selected VEC server  $j$ . On the other hand, when the result is not generated by the VEC server where the vehicle will arrive, the result handover is used to forward the result from VEC server  $j$  to the controller, and from the controller to the destination VEC server  $j^{\text{arr}}$  with which the vehicle will be attached. Note that  $j^{\text{arr}}$  can be estimated by (3). Therefore, the total delay of edge offloading can be obtained as:

$$T_i^j(t) = \frac{\mathcal{D}_i^{\text{in}}(t)}{r_{i,j}(t)} + \frac{2\mathcal{D}_i^{\text{in}}(t)}{r_f} \cdot \mathbb{I}_{(j \neq j^{\text{cur}})} + \frac{\mathcal{C}_i^{\text{req}}(t)}{f_{j,i}(t)} + \frac{2\mathcal{D}_i^{\text{out}}(t)}{r_f} \cdot \mathbb{I}_{(j \neq j^{\text{arr}})}. \quad (9)$$

*Cloud Offloading.* When task  $\mathcal{T}_i(t)$  is processed by cloud server  $o$ , the offloading delay mainly includes transmission delay, computation delay and vertical migration delay, i.e.,

$$T_i^o(t) = \underbrace{T_{i,j^{\text{cur}}}^{\text{tran}}(t)}_{\text{Transmission}} + \underbrace{T_{i,o}^{\text{comp}}(t)}_{\text{Computation}} + \underbrace{T_{j^{\text{cur}},o}^{\text{hand}}(t)}_{\text{Task handover}} + \underbrace{T_{o,j^{\text{arr}}}^{\text{hand}}(t)}_{\text{Result handover}}. \quad (10)$$

*i)  $T_{i,j^{\text{cur}}}^{\text{tran}}(t) = \frac{\mathcal{D}_i^{\text{in}}(t)}{r_{i,j}(t)}$*  is the delay for vehicle  $i$  to transmit the task to VEC server  $j^{\text{cur}}$  within the current service area. *ii)  $T_{i,o}^{\text{comp}}(t) = \frac{\mathcal{C}_i^{\text{req}}(t)}{f_{o,i}(t)}$*  is the computing delay of the task, wherein  $f_{o,i}(t)$  is the computation resources that is allocated to the task. *iii) similarly to edge offloading, the vertical migration delay consists of task handover delay and result handover delay, i.e.,  $T_{j^{\text{cur}},o}^{\text{hand}}(t) = \frac{\mathcal{D}_i^{\text{in}}(t)}{r_c}$  and  $T_{o,j^{\text{arr}}}^{\text{hand}}(t) = \frac{\mathcal{D}_i^{\text{out}}(t)}{r_c}$ , where  $r_c$  is the data rate between the VEC server and the cloud [35]. Therefore, the total delay of cloud offloading can be obtained as:*

$$T_i^o(t) = \frac{\mathcal{D}_i^{\text{in}}(t)}{r_{i,j}(t)} + \frac{\mathcal{C}_i^{\text{req}}(t)}{f_{o,i}(t)} + \frac{\mathcal{D}_i^{\text{in}}(t) + \mathcal{D}_i^{\text{out}}(t)}{r_c}. \quad (11)$$

*2) Energy Consumption:* Completing task  $\mathcal{T}_i(t)$  could incur additional costs for vehicle  $i$ , VEC server  $j$  or cloud server  $o$ .

*Local Offloading.* The energy consumption of vehicle  $j$  to execute task  $\mathcal{T}_i(t)$  locally is widely formulated as:

$$E_i^0(t) = \alpha_i \cdot (f_i(t))^{\tau-1} \cdot \mathcal{C}_i^{\text{req}}(t), \quad (12)$$

where  $\alpha_i \geq 0$  is the effective switched capacitance of vehicle  $i$ 's CPU that depends on the CPU chip architecture [59], and  $\tau > 0$  is the constant that is typically set as 2 or 3 [60].

*Edge/Cloud Offloading.* The energy consumption of server  $j \in \{\mathcal{E}, o\}$  to execute task  $\mathcal{T}_i(t)$  can be given as:

$$E_j^i(t) = \alpha_j \cdot (f_{j,i}(t))^{\tau-1} \cdot \mathcal{C}_i^{\text{req}}(t), \quad (13)$$

where  $\alpha_j \geq 0$  denotes the effective switched capacitance of server  $j$ 's CPU.

<sup>5</sup>The delay of result feedback is omitted for horizontal migration as mentioned in Section III-B.

#### D. Utility Model

When task  $\mathcal{T}_i(t)$  is completed by using the offloading strategy  $s_i^a(t)$  at time  $t$ , the utilities of vehicle  $i$ , VEC server  $j$ , and cloud server  $o$  are formulated as follows.

1) *Vehicle Utility*: The utility obtained by vehicle  $i \in \mathcal{V}$  from offloading task  $\mathcal{T}_i(t)$  is formulated as:

$$U_i^a(t) = w_i \cdot \Psi_i^a(t) - (1 - w_i)C_i^a(t), \quad (14)$$

where  $\Psi_i^a(t)$  and  $C_i^a(t)$  are the normalized satisfaction level of task completion delay and the normalized cost of task execution, respectively, when selecting offloading strategy  $a$ , and  $w_i$  is the weight coefficient of the satisfaction level.

First, the satisfaction function is a term that is widely used in economics, which is formulated as a logarithmic function that is convex and starts from zero. It has been employed to quantify the satisfaction level of task offloading [17], [27]. Based on these studies, the normalized satisfaction level can be calculated as:

$$\Psi_i^a(t) = \frac{\log(1 + (T_i^{\max} - T_i^a(t)))}{\log(1 + T_i^{\max})}, \quad (15)$$

where  $T_i^a(t)$  is the total delay for completing task  $\mathcal{T}_i(t)$ , which can be obtained based on (7), (9) and (11).

Second, the normalized cost of vehicle  $i$  can be computed based on the energy consumption for local offloading or the payment for remote offloading, i.e.,

$$C_i^a(t) = \begin{cases} \frac{E_i^0(t)}{E_i^{\max}}, & a = 0, \\ \frac{c_{j,i}(t) \cdot f_{j,i}(t)}{C_i^{\max}}, & a = j, j \in \{\mathcal{E}, o\}, \end{cases} \quad (16a)$$

$$(16b)$$

where  $E_i^0(t)$  (given by (12)) is the computation energy consumption of vehicle  $i$  and  $E_i^{\max}$  is the energy constraint of vehicle  $i$  [61],  $c_{j,i}(t)$  is the unit price of the computation resources charged by the VEC server or cloud server, and  $f_{j,i}(t)$  is the computation resources allocated to vehicle  $i$  by server  $j$ , and  $C_i^{\max}$  is the budget of vehicle  $i$  for the costs paid to the servers.

Therefore,  $U_i^a(t)$  can be obtained based on (12), (14) and (16) as:

$$U_i^a(t) = w_i \cdot \frac{\log(1 + (T_i^{\max} - T_i^a(t)))}{\log(1 + T_i^{\max})} - (1 - w_i) \left( \frac{\alpha_i \cdot (f_i(t))^{\tau-1} \cdot C_i^{\text{req}}(t)}{E_i^{\max}} \cdot \mathbb{I}_{(a=0)} + \frac{c_{j,i}(t) \cdot f_{j,i}(t)}{C_i^{\max}} \cdot \mathbb{I}_{(a=j)} \right). \quad (17)$$

2) *Server Utility*: The utility of server  $j \in \{\mathcal{E}, o\}$  obtained by executing task  $\mathcal{T}_i(t)$  is formulated as the revenue of task processing subtracting the cost of energy consumption:

$$U_j^i(t) = w_j \cdot \mathcal{R}_j^i(t) - (1 - w_j)C_j^i(t), \quad (18)$$

where  $\mathcal{R}_j^i(t)$  and  $C_j^i(t)$  are the normalized revenue and normalized cost of server  $j$ , respectively, and  $w_j$  is the weight coefficient of revenue.

First, the normalized revenue of server  $j$  to process task  $\mathcal{T}_i(t)$  can be given as:

$$\mathcal{R}_j^i(t) = \frac{c_{j,i}(t) \cdot f_{j,i}(t)}{C_j^{\max} \cdot f_j^{\max}}, \quad (19)$$

where  $C_j^{\max}$  is the maximum unit price of server  $j$ 's computation resources.

Second, the normalized energy consumption of server  $j$  to process task  $\mathcal{T}_i(t)$  can be given as:

$$C_j^i(t) = \frac{E_j^i(t)}{E_j^{\max}}, \quad (20)$$

where  $E_j^i(t)$  (given by (13)) is the computation energy consumption of server  $j$  and  $E_j^{\max}$  is the energy constraint of server  $j$ .

Therefore,  $U_j^i(t)$  can be obtained based on (18), (19), (20), and (13) as:

$$U_j^i(t) = w_j \cdot \frac{c_{j,i}(t) \cdot f_{j,i}(t)}{C_j^{\max} \cdot f_j^{\max}} - (1 - w_j) \cdot \frac{\alpha_j \cdot (f_{j,i}(t))^{\tau-1} \cdot C_j^{\text{req}}(t)}{E_j^{\max}}. \quad (21)$$

3) *Social Welfare*: Social welfare is employed in this work to quantify the system performance of computation resource allocation and task offloading for the servers and vehicles in the VEC network. Therefore, the social welfare at time  $t$  can be given as follows:

$$\begin{aligned} SW(t) &= \underbrace{\sum_{i \in \mathcal{V}} \sum_{a \in \mathcal{A}} p_i^{\text{gen}}(t) s_i^a(t) U_i^a(t)}_{\text{The total utility of vehicles}} \\ &\quad + \underbrace{\sum_{j \in \{\mathcal{E}, o\}} \sum_{i \in \mathcal{V}} s_j^i(t) p_i^{\text{gen}}(t) U_j^i(t)}_{\text{The total utility of servers}} \\ &= \sum_{i \in \mathcal{V}} \sum_{a \in \mathcal{A}} p_i^{\text{gen}}(t) \cdot s_i^a(t) \cdot (U_i^a(t) + U_a^i(t)). \quad (22) \end{aligned}$$

Note that (22) incorporates the local, edge, and cloud offloading strategies. If vehicle  $i$  offloads the task locally at time  $t$ , i.e.,  $a = 0$ , then the utility of server  $j$  to provide service for vehicle  $i$  is 0, i.e.,  $U_0^i(t) = 0$ .

#### IV. PROBLEM FORMULATION

The objective of this work can be transformed to maximize the social welfare over  $\mathbf{T}$  slots by jointly optimizing the task offloading strategy  $S_t = \{s_i^a(t)\}_{i \in \mathcal{V}, a \in \mathcal{A}, t \in \mathbf{T}}$ , and the computation resource allocation and pricing strategy  $S_c = \{f_j^i(t), c_{j,i}(t)\}_{j \in \{\mathcal{E}, o\}, i \in \mathcal{V}, t \in \mathbf{T}}$ . Therefore, JRATOP can be formulated as follows:

$$\mathbf{P} : \max_{S_t, S_c} \sum_{t=t_0}^{\mathbf{T}} SW(t)$$

$$\text{s.t. C1 : } s_i^a(t) \in \{0, 1\}, \forall i \in \mathcal{V}, a \in \mathcal{A}$$

$$\text{C2 : } \sum_{a \in \mathcal{A}} s_i^a(t) \leq 1, \forall i \in \mathcal{V}, a \in \mathcal{A}$$

$$\text{C3 : } p_i^{\text{gen}}(t) = \{0, 1\}, \forall i \in \mathcal{V}$$

$$\text{C4 : } s_i^a(t) \cdot T_i^a(t) \leq T_i^{\max}(t), \forall i \in \mathcal{V}, \forall j \in \{\mathcal{E}, o\}, a \in \mathcal{A}$$

$$\text{C5 : } s_i^a(t) \cdot T_{i,j}^{\text{tran}}(t) \leq T_{i,j}^{\text{soj}}, \forall i \in \mathcal{V}, \forall j^{\text{cur}} \in \{\mathcal{E}, o\},$$

$$\begin{aligned}
\text{C6} : s_i^a \cdot T_{j,j^{\text{arr}}}^{\text{hand}}(t) &\leq T_{i,j^{\text{arr}}}^{\text{soj}}, \forall i \in \mathcal{V}, \forall j \in \{\mathcal{E}, o\}, \forall a \in \{\mathcal{E}, o\} \\
\text{C7} : v^{\min} &\leq v_i(t) \leq v^{\max}, \forall i \in \mathcal{V} \\
\text{C8} : \sum_{i \in \mathcal{V}} s_i^j(t) \cdot f_{j,i}(t) &\leq f_j^{\max}, \forall j \in \{\mathcal{E}, o\} \\
\text{C9} : \sum_{i \in \mathcal{V}} s_i^j(t) &\leq N_j^{\text{core}}, \forall j \in \{\mathcal{E}, o\} \\
\text{C10} : s_i^0(t) \cdot E_i^0(t) &\leq E_i^{\max}, \forall i \in \mathcal{V} \\
\text{C11} : \sum_{i \in \mathcal{V}} s_i^j(t) \cdot E_j^i(t) &\leq E_j^{\max}, \forall j \in \{\mathcal{E}, o\} \\
\text{C12} : s_i^a(t) \cdot c_{a,i}(t) \cdot f_{a,i}(t) &\leq C_i^{\max}, \forall i \in \mathcal{V}, \forall a \in \{\mathcal{E}, o\}
\end{aligned} \tag{23}$$

where  $T_{i,j^{\text{cur}}}^{\text{soj}}$  and  $T_{i,j^{\text{arr}}}^{\text{soj}}$  can be obtained by (2), and  $j^{\text{arr}}$  can be obtained by (3). Constraints C1 and C2 are the values of offloading strategies of vehicles, which indicates that the vehicle can only select one strategy as its offloading decision. Constraint C3 represents each vehicle generates at mos time slot. Constraint C4 is the delay constraint of the task, which guarantees that the task is completed before the deadline. Constraint C5 ensures that the task uploading is completed before vehicle  $i$  moving out of the coverage of the connected VEC server  $j^{\text{cur}}$ . Constraint C6 guarantees that the result dispatch is completed before vehicle  $i$  moving out of the coverage of VEC server  $j^{\text{arr}}$  with which it will be attached. VEC server  $j^{\text{arr}}$  can be obtained by substituting  $T_i^{\text{move}} = T_{i,j^{\text{cur}}}^{\text{tran}}(t) + T_{j^{\text{cur}},j}^{\text{hand}}(t) \mathbb{I}_{(j \neq j^{\text{cur}})} + T_{i,j}^{\text{comp}}(t)$  into (3), where  $T_i^{\text{move}}(j \in \{\mathcal{E}, o\})$  is the moving duration before the result of task is generated. Constraint C7 poses constraints on the velocity of each vehicle. Constraints C8 and C9 limit the computation resources and the number of CPU cores, respectively, for each server. Constraints C10 and C11 constrain the energy budgets of vehicles and servers, respectively. Constraint C12 represents a vehicle's maximum payment for the computational resources provided by servers.

**Theorem 1.** The problem **P** formulated in (23) is NP-hard.

*Proof.* The detailed proof is given in Appendix A of the supplemental material, available online.

## V. BARGAIN-MATCH

To solve problem **P**, the algorithm of BARGAIN-MATCH is proposed for resource allocation and task offloading by using the bargaining and matching schemes. BARGAIN-MATCH mainly consists of the following two parts. i) For intra-server resource allocation, the bargaining game is used to stimulate the negotiation between the requesting vehicle and server on resource allocation and price in Section V-A. ii) For inter-server task offloading, the many-to-one matching is constructed between tasks and servers to stimulate both edge-edge collaboration for horizontal task migration and edge-cloud collaboration for vertical task migration.

### A. Resource Allocation and Pricing: A Bargaining Game-Based Scheme

1) *Fundamentals:* Denote the set of requesting vehicles that have tasks to upload at time  $t$  as  $\mathcal{V}^{\text{req}}(t) = \{i | i \in \mathcal{V}, p_i^{\text{gen}}(t) = 1, s_i^0(t) = 0\}$ . A finite-time bargaining game is constructed to stimulate the negotiation between the requesting vehicle and its expected server on the decisions of resource allocation and pricing in period  $\Delta t$ . The bargaining game is defined as a triplet of  $(\mathcal{P}_B, \mathcal{S}_B, \mathcal{U}_B)$ , which is as follows:

- $\mathcal{P}_B = \{i \in \mathcal{V}^{\text{req}}(t), j \in \{\mathcal{E}, o\}\}$  denotes the parties, i.e., a seller and a buyer. Vehicle  $i$  acts as a buyer that aims to offload its task by buying the required computation resources from the server. Moreover, server  $j$  acts as a seller who aims to execute the task of the vehicle by allocating the computation resources and charging from the vehicle.
- $\mathcal{S}_B = \{f_{j,i}(t), c_{j,i}(t)\}$  denotes the set of strategies. The strategy of vehicle  $i$  is to request the satisfactory amount of computation resources from the server, and the strategy of server  $j$  is to decide the satisfactory price of the computation resources that are sold to vehicle  $i$ .
- $\mathcal{U}_B = \{U_i^j(t), U_j^i(t)\}$  denotes the utilities of vehicle  $i$  and server  $j$ , wherein  $U_i^j(t)$  and  $U_j^i(t)$  are given in (17) and (21), respectively.

2) *Resource Allocation and Pricing:* In this section, the optimal resource allocation and pricing strategies are presented.

**Theorem 2.** For vehicle  $i$ , the expected optimal amount of computation resources it intends to request from target server  $j$  to offload task  $\mathcal{T}_i(t)$  is obtained as:  $f_{j,i}^*(t) = \frac{2w_i \cdot C_j^{\max}}{F(\text{var}_1, c_{j,i}(t)) - \log(1 + T_i^{\max}(t)) \cdot c_{j,i}(t) \cdot (1 - w_i)}$ , where  $F(\text{var}_1, c_{j,i}(t))$  is given by (24) shown at the bottom of next page, which is the function of variables  $\text{var}_1$  and  $c_{j,i}(t)$ .  $\text{var}_1 = \frac{D_i^{\text{in}}(t)}{r_{i,j}(t)} + \frac{2D_i^{\text{in}}(t)}{r_f} \cdot \mathbb{I}_{(j \neq j^{\text{cur}})} + \frac{2D_i^{\text{out}}(t)}{r_f} \cdot \mathbb{I}_{(j \neq j^{\text{arr}})}$  for edge server  $j \in \mathcal{E}$ , and  $\text{var}_1 = \frac{D_i^{\text{in}}(t)}{r_{i,j}(t)} + \frac{D_i^{\text{in}}(t) + D_i^{\text{out}}(t)}{r_c}$  for cloud server  $j = o$ .

*Proof.* The detailed proof is given in Appendix B of the supplemental material, available online.

**Lemma 1.** When the controller decides to offload task  $\mathcal{T}_i(t)$  to server  $j \in \{\mathcal{E}, o\}$  at time slot  $t$ , there exist a lower bound and an upper bound for the unit price of the computation resources that are allocated to the task, i.e.,  $c_{j,i}^{\min}(t) \leq c_{j,i}(t) \leq c_{j,i}^{\max}(t)$ , where  $c_{j,i}^{\min}(t) = \frac{(1 - w_j) \alpha_j \cdot (f_{j,i}(t))^{\tau-2} \cdot C_i^{\text{req}}(t) \cdot C_j^{\max} \cdot f_j^{\max}}{w_j \cdot E_j^{\max}}$  and  $c_{j,i}^{\max}(t) = \frac{w_i \log(1 + T_i^{\max} - T_i^j(t)) \cdot C_j^{\max}}{(1 - w_i) \cdot f_{j,i}(t) \cdot \log(1 + T_i^{\max})}$ .

*Proof.* The detailed proof is given in Appendix C of the supplemental material, available online.

According to Lemma 1, the bid-ask spread can be obtained as  $\Delta c_{j,i}(t) = c_{j,i}^{\max}(t) - c_{j,i}^{\min}(t)$ . The trade on the computation resources between vehicle  $i$  and VEC server  $j$  is modeled as the process of two players bargaining over the pie of size  $\Delta c_{j,i}(t)$  according to [62]. Obviously, both bargainers desire to reach an agreement on the proposal of the partition earlier because their utilities will be discounted over time. Therefore, the discount factor is introduced in the bargaining game to describe the discount of the partition in the future. The discount factor

captures the patience levels of the bargainers. In other words, the smaller discount factor indicates that the players are impatient with the delay of the negotiation. Accordingly, the discount factors of vehicle  $i$  and server  $j$  are formulated as follows:

$$\epsilon_i(t) = 1 - \frac{T_{i,j}^{\text{tran}}(t)}{T_i^{\text{max}}(t)}, \quad (25)$$

$$\epsilon_j(t) = 1 - \frac{T_{i,j}^{\text{comp}}(t)}{T_j^{\text{max}}(t)}. \quad (26)$$

For vehicle  $i$ , it is more impatient if it takes longer time to upload the task to the VEC server, leading to the lower value of  $\epsilon_i(t)$ . For VEC server  $j$ , it is more impatient if it takes longer time to execute the task. Besides, the larger task deadline  $T_i^{\text{max}}(t)$  indicates that the players have higher endurance to the delay, leading to higher values of  $\epsilon_i(t)$  and  $\epsilon_j(t)$ .

*Lemma 2.*

The bargaining game has an unique perfect partition. In the period  $T^b \in T(n)$  in which vehicle  $i$  makes a proposal, the optimal partitions are given as:

$$\begin{cases} \delta_i^{j*}(t) = \epsilon_i(t) - \frac{(1-\epsilon_i(t)) \left(1 - (\epsilon_i(t)\epsilon_j(t))^{\lceil \frac{T^b}{2} \rceil}\right)}{1 - \epsilon_i(t)\epsilon_j(t)}, \\ \delta_j^{i*}(t) = \frac{(1-\epsilon_i(t)) \left(2 - \epsilon_i(t)\epsilon_j(t) - (\epsilon_i(t)\epsilon_j(t))^{\lceil \frac{T^b}{2} \rceil}\right)}{1 - \epsilon_i(t)\epsilon_j(t)}. \end{cases}$$

In the period  $T^b \in T(n)$  when server  $j$  makes a proposal, the optimal partitions are given as:

$$\begin{cases} \delta_i^{j*}(t) = \frac{(1-\epsilon_j(t)) \left(1 - (\epsilon_i(t)\epsilon_j(t))^{\lceil \frac{T^b}{2} \rceil}\right)}{1 - \epsilon_i(t)\epsilon_j(t)}, \\ \delta_j^{i*}(t) = \frac{\epsilon_j(t)(1-\epsilon_i(t)) - (1-\epsilon_j(t))(\epsilon_i(t)\epsilon_j(t))^{\lceil \frac{T^b}{2} \rceil}}{1 - \epsilon_i(t)\epsilon_j(t)}. \end{cases}$$

*Proof.* The detailed proof is given in Appendix D of the supplemental material, available online.

*Theorem 3.* The optimal price of computation resource  $c_{j,i}^*(t)$  that the VEC server  $j$  charges vehicle  $i$  is obtained as: 1) in the period in which vehicle  $i$  makes a proposal,  $c_{j,i}^*(t) = c_{j,i}^{\text{max}}(t) - \Delta c_{j,i}(t) \cdot \delta_i^{j*}(t)$ ; 2) in the period in which VEC server  $j$  makes a proposal,  $c_{j,i}^*(t) = c_{j,i}^{\text{max}}(t) - \Delta c_{j,i}(t) \cdot \delta_j^{i*}(t)$ .

*Proof.* The detailed proof is given in Appendix E of the supplemental material, available online.

*Corollary 1.* It can be concluded that a deal on the optimal computation resource allocation  $f_{j,i}^*(t)$  and pricing  $c_{j,i}^*(t)$  can be achieved by (27) and (28), respectively, when  $f_{j,i}^*(t)$  and  $c_{j,i}^*(t)$  satisfy Theorems 2 and 3 simultaneously.

$$f_{j,i}^*(t) = \frac{2w_i \cdot C_i^{\text{max}}}{F(\text{var}_1, c_{j,i}^*(t)) - \log(1 + T_i^{\text{max}}(t)) \cdot c_{j,i}(t) \cdot (1 - w_i)}, \quad (27)$$

$$c_{j,i}^*(t) = \begin{cases} c_{j,i}^{\text{max}}(t) - \Delta c_{j,i}(t) \cdot \delta_i^{j*}(t), & (28a) \\ c_{j,i}^{\text{max}}(t) - \Delta c_{j,i}(t) \cdot \delta_j^{i*}(t). & (28b) \end{cases}$$

---

#### Algorithm 1: Resource Allocation and Pricing.

---

**Input:** Vehicle  $i$ , VEC server  $j$

**Output:** The optimal resource allocation and pricing strategy  $s_{j,i}^*$  between  $i$  and  $j$

- 1 **Initialization:**  $U_i^j(t) = 0$ ;  $U_j^i(t) = 0$ ;
  - 2 VEC server  $j$  sets the optimal allocation as  $f_{j,i}^*(t) = f_{j,i}^{\text{avl}}$ ;
  - 3 Calculate the optimal price  $c_{j,i}^*(t)$  based on Eq. (28);
  - 4 Calculate  $U_i^j(t)$  for vehicle  $i$  based on Eq. (17);
  - 5 Calculate  $U_j^i(t)$  for VEC server  $j$  based on Eq. (21);
  - 6 **if**  $U_i^j(t) > 0 \ \&\& \ U_j^i(t) > 0$  **then**  
     // An agreement is reached.
  - 7     **return**  $s_{j,i}^*(t) = (f_{j,i}^*(t), c_{j,i}^*(t))$ ;
  - 8 **else if**  $U_i^j(t) > 0 \ \&\& \ U_j^i(t) < 0$  **then**  
     Vehicle  $i$  proposes the optimal price  $c_{j,i}^*(t)$  based on Eq. (28a);
  - 10 **else if**  $U_i^j(t) < 0 \ \&\& \ U_j^i(t) > 0$  **then**  
     VEC server  $j$  proposes the optimal price  $c_{j,i}^*(t)$  based on Eq. (28b);
  - 12 **else**  
     Either of the players can propose the optimal price  $c_{j,i}^*(t)$  based on Eq. (28);
  - 14 **return**  $s_{j,i}^* = (f_{j,i}^*(t), c_{j,i}^*(t))$ ;
- 

Based on the result of optimal resource allocation and pricing, a negotiation approach between vehicle  $i$  and VEC server  $j$  that intend to conclude a transaction for task offloading is presented in Algorithm 1. The negotiation is mainly based on the pricing rule, which is defined as follows.

*Definition 1. Pricing Rule.* Before the transaction, VEC server  $j$  initially sets the optimal resource allocation  $f_{j,i}^*(t)$  as its available computation resource. Then, the resource price is updated based on the following rules.

- If  $U_i^j(t) > 0 \ \&\& \ U_j^i(t) > 0$ , an agreement on the optimal resource allocation and pricing is reached.
- If  $U_i^j(t) > 0 \ \&\& \ U_j^i(t) < 0$ , vehicle  $i$  proposes the optimal price based on (28a).
- If  $U_i^j(t) < 0 \ \&\& \ U_j^i(t) > 0$ , VEC server  $j$  proposes the optimal price based on (28b).
- If  $U_i^j(t) < 0 \ \&\& \ U_j^i(t) < 0$ , either VEC server  $j$  or vehicle  $i$  can propose the optimal price.

#### B. Offloading Strategy Selection: A Matching-Based Scheme

Denote the tasks that have not been decided where to offload as  $\mathcal{T}^{\text{req}}(t) = \{\mathcal{T}_i(t) | i \in \mathcal{V}\}$ . Then the offloading strategy of each task  $k \in \mathcal{T}^{\text{req}}(t)$  is decided using a many-to-one matching scheme. Specifically, if task  $k$  is matched with VEC server  $j \in \mathcal{E}$ , it will be offloaded horizontally from VEC server  $j^{\text{cur}}$  to VEC

---


$$F(\text{var}_1, c_{j,i}(t)) = \sqrt{\frac{c_{j,i}(t) \log(1 + T_i^{\text{max}}(t)) (1 - w_i) (c_{j,i}(t) \cdot C_i^{\text{req}}(t) \cdot \log(1 + T_i^{\text{max}}(t)) (1 - w_i) + 4C_i^{\text{max}} w_i (1 + T_i^{\text{max}}(t) - \text{var}_1))}{C_i^{\text{req}}(t)}} \quad (24)$$

server  $j$ ; if it is matched with server  $o$ , it will be offloaded vertically from VEC server  $j^{\text{cur}}$  to the cloud.

1) *Fundamentals*: The matching is described as a triplet of  $(\mathcal{P}_M, \Omega, \Phi)$ :

- $\mathcal{P}_M = (\mathcal{T}^{\text{req}}(t), \{\mathcal{E}, o\})$  denotes two disjoint sets of players where  $\mathcal{T}^{\text{req}}(t) = \{\mathcal{T}_i(t) | i \in \mathcal{V}\}$  is the set of tasks that have not been decided where to offload currently, and  $\{\mathcal{E}, o\}$  is the set of servers.
- $\Omega = (\Omega_k^j, \Omega_j^k)$  denotes the preference lists of the tasks and servers. Each task  $k \in \mathcal{T}^{\text{req}}(t)$  has a descending ordered preferences on the servers, i.e.,  $\Omega_k^j = \{j | j \in \{\mathcal{E}, o\}, j \succ_i j'\}$ , where  $\succ_k$  denotes the preference of task  $k$  towards the servers. Furthermore, each server  $j \in \{\mathcal{E}, o\}$  has a descending ordered preference list over the tasks, i.e.,  $\Omega_j^k = \{k \in \mathcal{T}^{\text{req}}(t), k \succ_j k'\}$ .
- $\Phi \subseteq \{k | k \in \mathcal{T}^{\text{req}}(t)\} \times \{\mathcal{E}, o\}$  is the many-to-one matching between the tasks and servers. Each task  $k \in \mathcal{T}^{\text{req}}(t)$  can be matched with at most one server, i.e.,  $\Phi(k) \in \{\mathcal{E}, o\}$ , and each server  $j \in \{\mathcal{E}, o\}$  can be matched with multiple tasks, i.e.,  $\Phi(j) \subseteq \{k | k \in \mathcal{T}^{\text{req}}(t)\}$ .

2) *Preference List Construction*: The preference lists are constructed as follows.

- Predict the optimal resource allocation  $f_{j,i}^*(t)$  and price  $c_{j,i}^*(t)$  allocated by server  $j$  to each task  $k = \mathcal{T}_i(t) \in \mathcal{T}^{\text{req}}(t)$  based on Algorithm 1. Note that  $k = \mathcal{T}_i(t)$  means that task  $k$  is generated by vehicle  $i$  at time  $t$ .
- Calculate the values of preference for each server  $j \in \{\mathcal{E}, o\}$  on tasks  $k \in \mathcal{T}^{\text{req}}(t)$  as:

$$\rho_j(k) = U_j^i(t), \quad k = \mathcal{T}_i(t), \quad (29)$$

where  $U_j^i(t)$  is the utility of server  $j$  to process task  $k$  of vehicle  $i$  (in (18)).

- Construct the matching list for each server  $j \in \{\mathcal{E}, o\}$  by ranking the preference values as a descending order:

$$\rho_j(k) > \rho_j(k') \Leftrightarrow k \succ_j k', \quad \Omega_j^k = \{k, k'\} \quad (30)$$

- Calculate the preference values of each task  $k \in \mathcal{T}^{\text{req}}(t)$  on each server  $j \in \{\mathcal{E}, o\}$  as:

$$\rho_k(j) = U_i^j(t), \quad k = \mathcal{T}_i(t) \quad (31)$$

where  $U_i^j(t)$  is the utility obtained by vehicle  $i$  when its task  $k$  is processed by server  $j$  at time  $t$ .

- Construct the matching list for each task  $k \in \mathcal{T}^{\text{req}}(t)$  by ranking the preference values as a descending order:

$$\rho_k(j) > \rho_k(j') \Leftrightarrow j \succ_i j', \quad \Omega_k^j = \{j, j'\}. \quad (32)$$

3) *Matching Construction*: The matching scheme is implemented as follows.

- The rejected set is initialized as  $\mathcal{T}^{\text{rej}}(t) = \mathcal{T}^{\text{req}}(t)$ .
- Select the most preferred server  $j' = \Omega_k^j[0]$  for each task  $k \in \mathcal{T}^{\text{rej}}$ , and adds  $j'$  to the matching list of each task temporarily:

$$\Phi(k) = \Phi(k) \cup j'. \quad (33)$$

---

**Algorithm 2: Matching Algorithm for Tasks and Servers in Time Slot  $t$ .**


---

**Input:** The requesting vehicle set  $\mathcal{V}^{\text{req}}(t)$ , task set  $\mathcal{T}^{\text{req}}(t) = \{\mathcal{T}_i(t) | i \in \mathcal{V}^{\text{req}}(t)\}$ , and server set  $\{\mathcal{E}, o\}$

**Output:** The optimal matching list  $\Phi(t)$ , the offloading strategy  $S_{\text{off}}^*(t)$ , and the resource allocation strategy  $S_{\text{all}}^*(t)$ .

```

1 Initialization:  $\mathcal{T}^{\text{rej}}(t) = \mathcal{T}^{\text{req}}(t)$ ,  $\mathcal{E}' = \emptyset$ ,  $\Phi^* = \emptyset$ ;
  // Preference lists construction
2 for  $k \in \mathcal{T}^{\text{req}}(t)$  do
3   for  $j \in \{\mathcal{E}, o\}$  do
4     Call Algorithm 1 for  $s_{j,i}^*(t) = (f_{j,i}^*(t), c_{j,i}^*(t))$ ;
5     Calculate the preference values of server  $j$  on
      task  $k$  as Eq. (29);
6     Construct the matching list of server  $j$  based
      on Eq. (30);
7     Calculate the preference values of task  $k$  on
      server  $j$  as Eq. (31);
8     Construct the matching list of task  $k$  as Eq.
      (32);
9   end
10 end
  // Matching construction
11 while There exists do
12   for task  $k = \mathcal{T}_i(t) \in \mathcal{T}^{\text{rej}}(t)$  do
13     Select the most preferred server  $j'$ ;
14     Update the preference list of  $k$  as Eq. (33);
15     Update the preference list of  $j'$  as Eq. (34);
16   end
17   for server  $j \in \{\mathcal{E}', o\}$  that receives new requests
      do
18     Update matching list based on Eq. (35);
19     Update  $\mathcal{T}^{\text{rej}}(t)$  based on (36);
20     if  $D_j \neq \emptyset$  then
21       for task  $k \in D_j$  do
22         Update the preference list as Eq. (37a);
23         Update the matching list as Eq. (37b);
24       end
25     end
26   end
27 end
28 return  $\Phi(t)$ ,
     $S_{\text{off}}^*(t) = \{s_i^{a*}(t) | a^* = \Phi^*(k), k = \mathcal{T}_i(t) \in \mathcal{T}^{\text{req}}(t)\}$ ,
     $S_{\text{all}}^*(t) = \{s_{j,i}^*(t) | j \in \{\mathcal{E}, o\}, k = \mathcal{T}_i(t) \in \Phi^*(j)\}$ ;

```

---

- If task  $k$  prefers server  $j' \in \{\mathcal{E}, o\}$ , add task  $k$  to the matching list of server  $j'$  temporarily:

$$\Phi(j') = \Phi(j') \cup k. \quad (34)$$

- Update the matching list  $\Phi(j)$  of each server  $j \in \{\mathcal{E}, o\}$  by remaining the top- $n$  most preferred tasks and removing

the less preferred tasks:

$$\begin{cases} |\Phi(j)| \leq n \leq N_j^{\text{core}}, \\ \sum_{k \in \Phi(j)} f_{j,i}^*(t) \leq f_j^{\text{max}}, k = \mathcal{T}_i(t), \end{cases} \quad (35a)$$

$$\Phi(j) = \Phi(j) \setminus D_j \quad (35b),$$

where  $n$  is the number of server  $j$ 's CPU cores that are idle at the current time,  $N_j^{\text{core}}$  is the total number of server  $j$ 's CPU cores, and  $D_j$  is the set of less preferred tasks of server  $j$ .

e) Add the tasks in  $D_j$  to the rejected set:

$$\mathcal{T}^{\text{rej}}(t) = \mathcal{T}^{\text{rej}}(t) \cup D_j. \quad (36)$$

f) Update the preference list and matching list of each deleted task  $k \in D_j$ :

$$\begin{cases} \Omega_k^j = \Omega_k^j \setminus \{j\}, \\ \Phi(k) = \Phi(k) \setminus \{j\}, \end{cases} \quad (37a)$$

$$(37b)$$

g) For the tasks  $k = \mathcal{T}_i(t) \in \mathcal{T}^{\text{rej}}(t)$  that are deleted in the last iteration, repeat the steps b) to d) until all tasks have been matched with a server, or the unmatched tasks have been rejected by all servers.

### C. Main Steps of BARGAIN-MATCH and Analysis

In this section, the main steps of BARGAIN-MATCH is shown in Algorithm 3, and the corresponding stability, optimality, and computational complexity are presented.

1) *Stability: Definition 2.* Blocking pair. Assuming that  $k \in \mathcal{T}^{\text{req}}(t)$  and  $j \in \{\mathcal{E}, o\}$  are not matched with each other under matching  $\Phi$ , i.e.,  $i \neq \Phi(j)$  and  $j \neq \Phi(i)$ ,  $\Phi$  is blocked by the blocking pair  $(i, j)$  if and only if  $i$  and  $j$  prefer each other to  $j' = \Phi(i)$  and  $i' = \Phi(j)$ , respectively.

*Definition 3.* Stable matching. A matching is stable if and only if there exists no blocking pair [63].

*Theorem 4.* The matching  $\Phi$  proposed by this work is stable for every  $k \in \mathcal{T}^{\text{req}}(t)$  and  $j \in \{\mathcal{E}, o\}$ .

*Proof.* The detailed proof is given in Appendix F of the supplemental material, available online.

2) *Optimality: Theorem 5.* The matching  $\Phi$  proposed by this work is weak Pareto optimal for each  $k \in \mathcal{T}^{\text{req}}$  and  $j \in \{\mathcal{E}, o\}$ .

*Proof.* The detailed proof is given in Appendix G of the supplemental material, available online.

3) *Complexity Analysis: Theorem 6.* BARGAIN-MATCH has a polynomial worst-case complexity in each time slot, i.e.,  $\mathcal{O}((|\mathcal{E}| + 1) \cdot (2|\mathcal{V}| + \min\{|\mathcal{E}| + 1, |\mathcal{V}|\}))$ , where  $|\mathcal{V}|$  and  $|\mathcal{E}|$  are the number of vehicles and VEC servers, respectively.

*Proof.* The detailed proof is given in Appendix H of the supplemental material, available online.

## VI. SIMULATION RESULTS AND ANALYSIS

### A. Simulation Setup

In this section, the proposed approach is evaluated by simulations implemented in MATLAB(R) 9.9 (R2020b) in a 2.70 GHz Intel Core i7 processor. The road scenario is generated by the *Automated Driving Toolbox* 3.2, where 30 VEC servers are placed on a 10 km 6-lane bidirectional road and 100 vehicles are

### Algorithm 3: BARGAIN-MATCH.

---

**Input:**  $\mathcal{V}, \mathcal{E}, \mathbf{T}$   
**Output:**  $SW$

- 1 **Initialization:** Initialize time  $t = 0$ , social welfare  $SW = 0$ , initial positions of vehicles  $\mathbf{P}_i$ ,  $i \in \mathcal{V}$ , and positions of servers  $\mathbf{P}_j$ ,  $j \in \{\mathcal{E}\}$ ;
- 2 **while**  $t \leq \mathbf{T}$  **do**
- 3     Create a list  $\mathcal{T}^{\text{req}}(t)$  for the tasks that have not been determined where to be offloaded;
- 4     Call Algorithm 2 to obtain  $\Phi^*(t)$ ,  $S_{\text{off}}^*(t)$ , and  $S_{\text{all}}^*(t)$ ;
- 5     **if**  $\Phi \neq \emptyset$  **then**
- 6         **for**  $j \in \Phi$  **do**
- 7             // Resource trading
- 8             Allcate resource  $f_{j,i}^*(t)$  to  $i$ ;
- 9             Charge  $c_{j,i}^*(t)$  for per unit of resource on vehicle  $i$ ;
- 10            // Task offloading
- 11            Add the task  $k \in \Phi(j)$  to the task list of server  $j$ ;
- 12            // social welfare calculation
- 13            Calculate the current social welfare  $SW(t)$  based on Eq. (22);
- 14            // State update
- 15            Update the task processing list of server  $j$ ;
- 16            Update the available computation resources of server  $j$ ;
- 17         **end**
- 18     **end**
- 19     Calculate the social welfare  $SW = SW + SW(t)$ ;
- 20     **if**  $t \% T_0 == 0$  **then**
- 21         Update the mobility of vehicles;
- 22     **end**
- 23     Update time  $t = t + \Delta t$ ;
- 24 **end**
- 25 **return**  $SW$ ;

---

randomly located on the road initially. Moreover, the vehicles run at the speed ranges of [2, 30] m/s in either direction. The default values of the simulation parameters are listed in Table II.

This work evaluates the proposed BARGAIN-MATCH in comparison with three benchmark schemes, i.e., the *entire local offloading (ELO)*, *exhaustive offloading (EXO)*, *nearest VEC offloading (NVO)*, and *entire cloud offloading (ECO)*. Besides, the *non-cooperative game-based offloading (NCO)* [16] and *One-to-one matching and price-rising-based offloading and resource allocation (OPORA)* [26] are tailored to be suited to the approach in this paper since there is no feasible solution that can be directly applied to this problem. These approaches are described as follows.

- ELO: all vehicles execute their tasks locally.
- EXO: the tasks of each vehicle are exhaustively offloaded using the optimal offloading strategy.
- NVO: the tasks of each vehicle are offloaded to the nearest VEC server with which the vehicle is currently attached.

TABLE II  
SIMULATION PARAMETERS

Symbol	Meaning	Default value
$\alpha$	CPU parameters	$7.8^{-21}$ [60]
$B_{i,j}$	Bandwidth between vehicle $i$ and VEC server $j$	40 MHz [17]
$\beta^L/\beta^{NL}$	Path loss exponent for LoS/NLoS communication	3/4 [64]
$c$	Speed of light	$3 \times 10^8$ m/s
$C_i^{\max}$	The budget of vehicle $i$ for the costs paid to the servers	20 \$
$C_j^{\max}$	The maximum unit price of server $j$ 's computation resources	$1 (\$/\text{GHz}) \cdot f_j^{\max}$
$C_i^{\text{req}}(t)$	Required computation resources of each bit	[500, 1500] cycles/bit [65]
$d_0$	Reference distance	1 m
$\mathcal{D}_i^{\text{in}}(t)$	Task size	[400, 1000] KB [17]
$\mathcal{D}_i^{\text{out}}(t)$	Task result	[0.1, 1] KB [66]
$E_i^{\max}$	Energy constraint of vehicle $i$	$1 (\text{Wh}/\text{GHz}) \cdot f_i^{\max}$ [61]
$E_j^{\max}$	Energy constraint of server $j$	$1 (\text{Wh}/\text{GHz}) \cdot f_j^{\max}$ [61]
$f_c$	Carrier frequency	5.9 GHz [67]
$f_i^{\max}$	Computation resources of vehicle $i$	[0.5, 1] GHz [68]
$f_j^{\max}$	Computation resources of VEC server $j$	[2, 10] GHz [17]
$f_o^{\max}$	Computation resources of cloud server $o$	30 GHz [20]
$m^L/m^{NL}$	Nakagami fading parameter for LoS/NLoS communication	2/1 [64]
$N_i^{\text{core}}$	The CPU core of vehicle $i$	1
$N_j^{\text{core}}$	The CPU core of VEC server $j$	[2, 8]
$N_o$	Noise power	-98 dBm
$P_i(t)$	Transmit power	[-85, 44.8] dBm [67]
$r_f$	Data rate of fiber link	4 Gb/s [69]
$r_c$	Data rate between the edge and cloud	100 Mb/s [35]
$\theta^L/\theta^{NL}$	Standard deviation of shadowing for LoS/NLoS communication	3 dB/4 dB [57]
$\tau$	CPU parameters	3 [60]
$T_i^{\max}(t)$	The maximum permissible delay	[0.1, 5] s [70]
$w_i/w_j$	The weight coefficient of vehicle $i$ /server $j$	[0, 1]

- ECO: all tasks are offloaded to the cloud server.
- NCO: each vehicle competitively decides the optimal offloading probability by playing a distributed non-cooperative game. Since NCO is designed for the single-VEC server scenario and single-time decision in [16], it is adjusted in this paper to adapt to the multi-VEC server and period-time scenario.
- OPORA: each task of the vehicle is assigned to a VEC server based on the one-to-one matching scheme, and each VEC server is stimulated to allocate the resource using the price-rising scheme.

### A. System Performance

In this section, we evaluate the impacts of different system parameters on the performance of social welfare, vehicle utility, and server utility.

1) *Effect of Time*: Fig. 2(a), (b), and (c) show the comparative results of the social welfare, the total utility of vehicles, and the total utility of VEC servers among the seven algorithms in terms of time. It can be observed from Fig. 2 that as time elapses, the social welfare, total utility of vehicles and total utility of servers for all schemes increase. This is because more tasks of vehicles are executed successfully along with time. Moreover, it can be observed that BARGAIN-MATCH outperforms ELO, EMO, NVO, NCO and OPORA in terms of social welfare, vehicle utility and server utility. The reasons are as follows. First, ELO, EMO, NVO and NCO mainly focus on optimizing the task offloading strategy for vehicles but do not consider the resource allocation strategy for VEC servers. The entire local offloading of ELO, the exhaustive offloading of EXO, the nearest VEC server offloading of NVO, the entire cloud offloading of ECO, and the competitive offloading of NCO could lead to congestion and resource over-use at certain vehicles or VEC servers. Furthermore, although OPORA achieves superior VEC utility than ELO, EMO, NVO and NCO schemes due to the price incentive strategy, it is inferior to BARGAIN-MATCH because it adopts the one-to-one matching strategy and the incentive of random raising pricing, which are less efficient compared to the many-to-one matching strategy and the bargaining incentive. On the one hand, the many-to-one matching of BARGAIN-MATCH improves both the amount of offloaded tasks and the utilization of computation resource by horizontally or vertically offloading the tasks to the VEC servers or cloud server. On the other hand, the bargaining incentive scheme of BARGAIN-MATCH facilitates the negotiation between the servers and requesting vehicles on the optimal decisions of computation resource allocation and pricing. Consequently, this set of simulation results shows that BARGAIN-MATCH has the overall superior performance on social welfare, vehicle utility, and VEC server utility among the seven algorithms.

2) *Effect of Vehicle Numbers*: Fig. 3(a), (b), and (c) compare the impact of the number of vehicles on the performance of social welfare, the total utility of vehicles, and the total utility of servers for different schemes. First, ELO shows the worst performance in terms of social welfare, vehicle utility, and server utility, which is mainly because all tasks are executed locally on vehicles. Furthermore, the social welfare, vehicle utility, and server utility of NVO and ECO exhibit initial upward and then downward tendencies as the number of vehicles grows. This is mainly due to the aggregated amount of tasks with increasing vehicles, which could lead to the possible overload of the nearest VEC servers when adopting EVO and the increasing unfulfilled tasks when adopting ECO. Moreover, with the increase of vehicles, the social welfare and vehicle utility for OPORA show the initial increase and subsequent decrease trends, and the server utility for it rises at a diminishing rate, which is mainly due to the one-to-one offloading strategy and the random increasing price incentive. Additionally, NCO has some random fluctuations in social welfare and vehicle utility and increases slightly in server utility with increasing vehicles. Although the variation tendency of NCO is not statistically significant, it shows inferior performance compared to most of the other schemes (except for ELO). This could be attributed to the probabilistic offloading

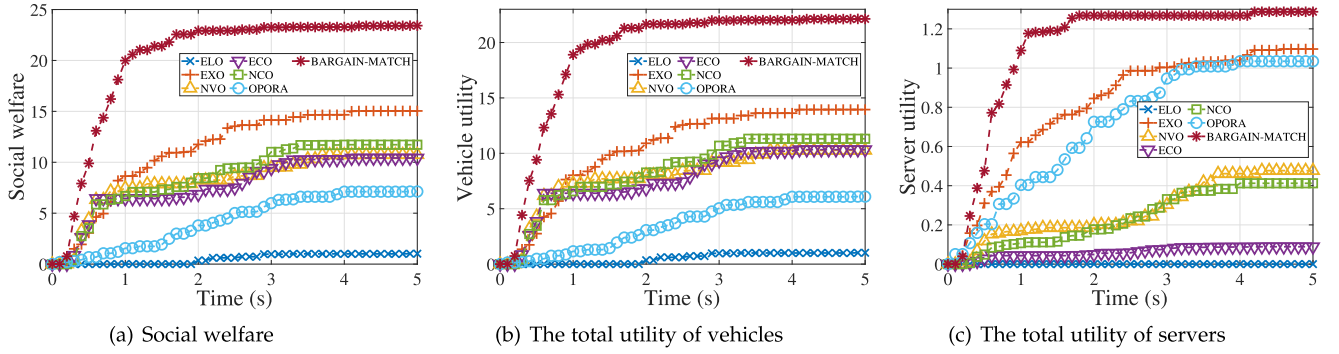


Fig. 2. System performance with respect to time. (a) Social welfare. (b) Vehicle utility. (c) Server utility.

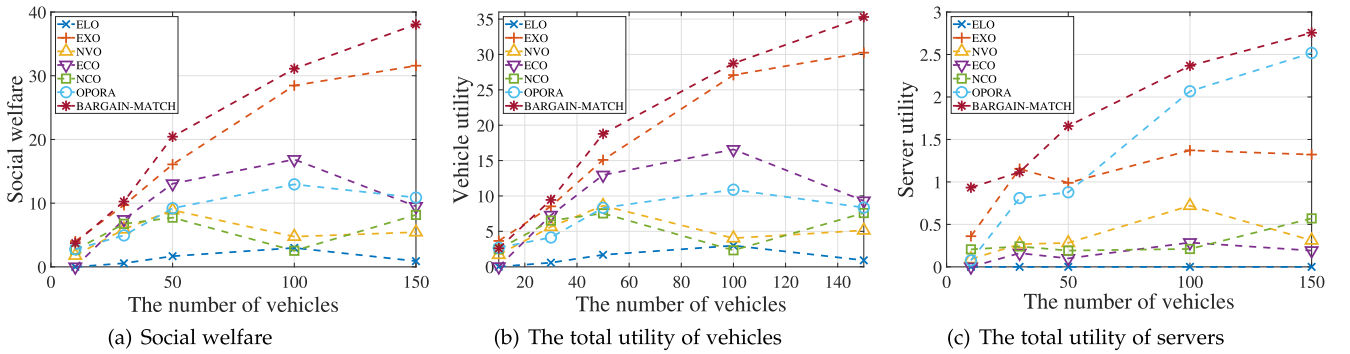


Fig. 3. System performance with respect to time. (a) Social welfare. (b) Vehicle utility. (c) Server utility.

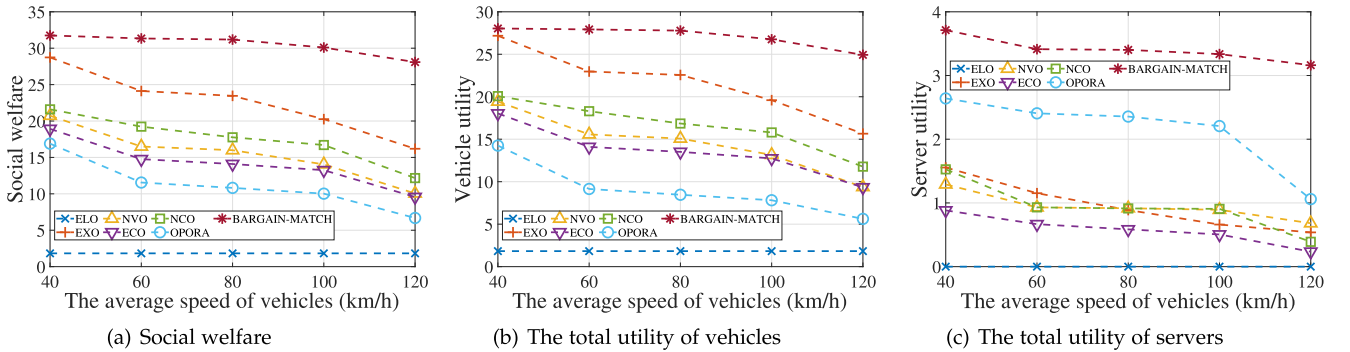


Fig. 4. System performance with respect to the average speed of vehicles. (a) Social welfare. (b) Vehicle utility. (c) Server utility.

strategy and the increased competition among vehicles. Besides, EXO shows initial increasing trends in social welfare, vehicle utility, and server utility with the increasing number of vehicles, and the trends slow down gradually or exhibit a decreasing trend. As explained before, although tasks can be offloaded to the optimal server, the exhaustive offloading strategy of EXO could lead to possible congestion or resource shortage for certain VEC servers, as the vehicles continuously increase. Last, it can be observed that BARGAIN-MATCH exhibits progressively increasing trends in the performance of social welfare, vehicle utility, and server utility, maintaining a relatively superior level among the seven schemes. The set of simulation results indicates

the better scalability of the proposed BARGAIN-MATCH with an increasing number of vehicles.

3) *Effect of Vehicle Speed*: Fig. 4(a), (b), and (c) compare the social welfare, the total utility of vehicles, and the total utility of servers among the comparative algorithms, respectively. First, it can be observed from Fig. 4 that the ELO shows the invariant but the worst performance in terms of social welfare, vehicle utility, and server utility with respect to the average speed of vehicles. This is obvious since the tasks of vehicles are executed locally without communicating with the VEC servers. Furthermore, for the algorithms of EXO, NVO, ECO, NCO, OPORA and BARGAIN-MATCH, the curves of social welfare, vehicle

utility, and server utility show overall downward trends with the increasing speed of vehicles. This is mainly because the high mobility of vehicles indicates that the vehicles could move out of the service range of the VEC server during the task uploading or task computing more frequently, leading to more repetitive handovers or even service interruptions. Therefore, the high mobility of vehicles could cause increased service delay and more task failures, which further results in the degraded satisfaction level of vehicles and the decreased revenues of servers. Specifically, the performances of EXO, NVO, ECO, NCO, and OPORA degrade significantly when the average speed exceeds 100 km/h. Finally, it can be observed that the performances of the proposed BARGAIN-MATCH exhibit relatively steady downward trends as the average speed of vehicles increases, maintaining superior levels among the seven schemes. The reason is that the proposed BARGAIN-MATCH performs cooperative decisions of task offloading by considering both horizontal and vertical task migrations, where the handover delay is incorporated to improve the connectivity for moving vehicles. In conclusion, this set of simulation results demonstrates that the proposed BARGAIN-MATCH can achieve relatively stable and superior performance in terms the social welfare, vehicle utility, and server utility against varying speeds of vehicles.

4) *Effect of Task Size:* The effect of the initial price of the computation resources on the system performance for the comparative algorithms is presented in Appendix I.1.1 of the supplementary material, available online, due to the page limitation.

5) *Effect of the Computation Resources of VEC Servers:* The effect of the initial price of the computation resources on the system performance for the comparative algorithms is presented in Appendix I.1.2 of the supplementary material, available online, due to the page limitation.

6) *Effect of the Initial Price of Computation Resources:* The effect of the initial price of the computation resources on the system performance for the comparative algorithms is presented in Appendix I.1.3 of the supplementary material, available online, due to the page limitation.

### C. System Efficiency

For performance evaluation, the following statistics are collected: the generation time of each task  $t_i^{\text{req}}(t)$ ,  $\forall i \in \mathcal{V}, t \in \mathbf{T}$ ; the successful completion time of each task  $t_i^{\text{com}}(t)$ ,  $\forall i \in \mathcal{V}, t \in \mathbf{T}$ ; the number of successfully completed tasks during the considered timeline, which is denoted as  $N^{\text{succ}}$ . Based on these statistics, the following performance metrics are defined.

- *Average processing rate (APR)* is defined as the average amount of task (in bits) that is processed per unit time, which is given as:

$$\text{APR} = \frac{\sum_{t \in \mathbf{T}} \sum_{i \in \mathcal{V}} p_i^{\text{gen}}(t) \cdot C_i^{\text{req}}(t)}{\sum_{t \in \mathbf{T}} \sum_{i \in \mathcal{V}} p_i^{\text{gen}}(t) \cdot (t_i^{\text{cmp}}(t) - t_i^{\text{req}}(t))}. \quad (38)$$

- *Average completion delay (ACD)* is defined as the average delay of completing a task successfully:

$$\text{ACD} = \frac{\sum_{t \in \mathbf{T}} \sum_{i \in \mathcal{V}} p_i^{\text{gen}}(t) \cdot (t_i^{\text{cmp}}(t) - t_i^{\text{req}}(t))}{\sum_{t \in \mathbf{T}} \sum_{i \in \mathcal{V}} p_i^{\text{gen}}(t)}. \quad (39)$$

- *Average completion ratio (ACR)* is defined as the ratio of tasks that are successfully completed to the total number of tasks generated during the considered duration, which is as follows:

$$\text{ACR} = \frac{N^{\text{succ}}}{\sum_{t \in \mathbf{T}} \sum_{i \in \mathcal{V}} p_i^{\text{gen}}(t)}. \quad (40)$$

1) *Effect of Task Size:* Fig. 5(a), (b), and (c) compare the APR, ACD, and ACR of the seven algorithms under different task sizes, respectively. First, both the APR and ACD of ELO, EXO, NVO, ECO, and NCO show overall upward trends with the increasing of task size, and the ACR of them show the opposite trends with the task size increases. Obviously, this is because the workloads of vehicles or servers become heavier with the increasing of task sizes, leading to the increased amount of task processing per unit time, the increased delay of task completion, and the decreased task completion ratio. Besides, it can be observed that the proposed BARGAIN-MATCH shows a significant rising trend in APR, a slight upward trend in ACD, and a slight downward trend in ACR with the increasing of task size. This implies that the processing rate of BARGAIN-MATCH increases significantly with relatively low costs of delay and task failure as the workload increases. Furthermore, BARGAIN-MATCH achieves the highest APR, the lowest ACD, and the highest ACR compared to the other schemes with the increasing of task size. BARGAIN-MATCH tries to stimulate cooperation among servers for inter-server task offloading and cooperation between servers and vehicles for intra-server resource allocation according to the varying works and the available resources of servers. In conclusion, the result set in Fig. 5 demonstrates the efficiency of BARGAIN-MATCH in terms of the APR, ACD, and ACR under varying task sizes.

2) *Effect of the Computation Resources of VEC Servers:* The effect of the computation resources on the system efficiency for the comparative algorithms is given in Appendix I.2.1 of the supplementary material, available online, due to the page limitation.

3) *Effect of the Initial Price of Computation Resources:* The effect of the initial price of the computation resources on the system efficiency for the comparative algorithms is given in Appendix I.2.2 of the supplementary material, available online, due to the page limitation.

### D. Algorithm Running Time

To evaluate the execution time of different approaches, Fig. 6 shows the average running time versus the number of vehicles for the seven algorithms. To show the impact of the processors, the proposed approach is implemented on a relative strong device ( $D_s$ ) equipped with Intel Core i7-12700H, 2.70 GHz processor, 16.0 GB RAM memory and on a relative weak device ( $D_w$ ) equipped with Intel Core i7-9750H, 2.60GHz processor, 16.0 GB RAM memory, respectively. From the perspective of the comparative approaches, it can be observed from Fig. 6 that the average running time increases for each algorithm with increasing number of vehicles. Specifically, the algorithms of ELO, NVO, ECO, and NCO have lower time complexity

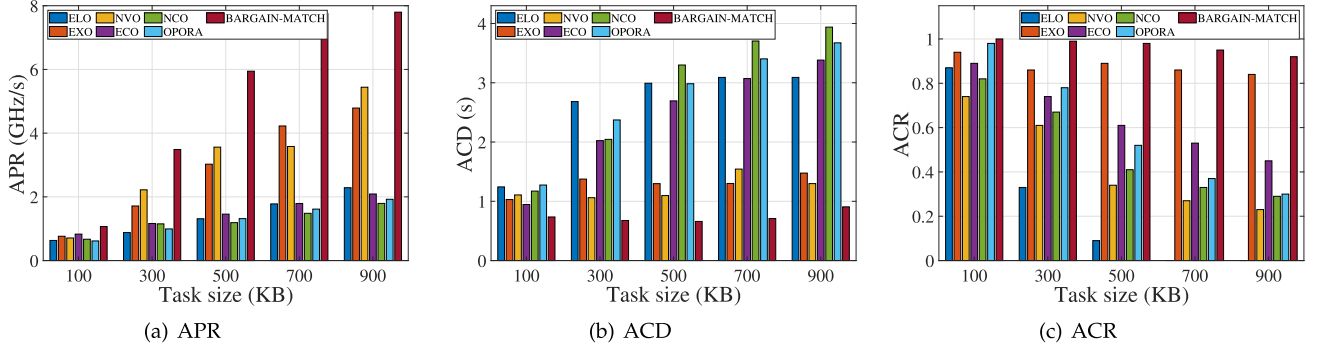


Fig. 5. System efficiency with respect to the average size of tasks. (a) APR. (b) ACD. (c) ACR.

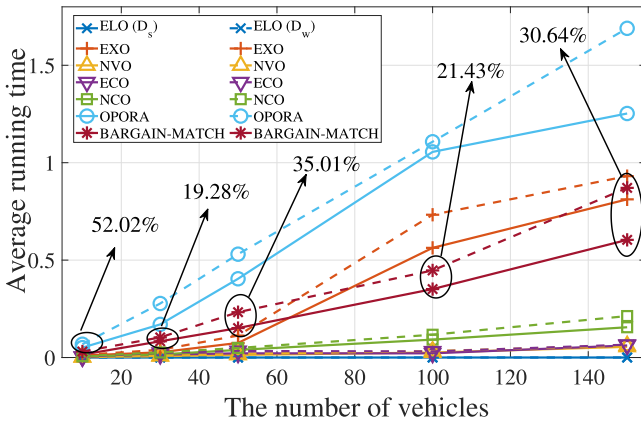


Fig. 6. Average running time with respect to the number of vehicles.

compared to EXO, OPOPRA, and BARGAIN-MATCH. This is because these approaches make offloading decisions directly, which however have relative inferior system performance and efficiency compared to the other approaches, as shown in Figs. 2, 3, 4, and 5. Furthermore, OPOPRA shows significantly higher time complexity among the seven schemes, which is mainly attributed to the time-consuming strategies of one-to-one matching and the random price rising incentive. Moreover, it can be observed that the average execution time of BARGAIN-MATCH increases linearly with increasing number of vehicles, which is consistent with the theoretical analysis. Besides, EXO takes less execution time than BARGAIN-MATCH when the number of vehicles is relative small ( $\leq 75$ ), while it is more time-consuming when the network becomes denser, and the difference enlarges gradually. From the perspective of the capability of devices, it can be observed that the average running time decreases approximately 19.28% to 52.02% when the simulation runs on  $D_s$ . Concluding from Fig. 6, BARGAIN-MATCH achieves superior performance in social welfare, vehicle utility, and server utility with the time complexity higher than the schemes that adopt direct decision making and lower than EXO in relative dense networks. Furthermore, the proposed approach can be completed in polynomial time with linearly increased complexity over the number of vehicles. Besides, the execution time of the proposed approach could be further reduced when it runs in the real VEC

node with stronger processing capability than the device we used.

## VII. DISCUSSION

### A. The Case of Vehicular Applications

Based on the standards of 5G Automotive Association (5GAA) [71] and ETSI MEC [72], [73], we consider the following vehicular applications, i.e., i) *vehicle collision warning*, ii) *emergency break warning*, iii) *traffic jam warning*, iv) *hazardous location warning*, and v) *speed harmonization*. The characteristics of each application that are mapped to the task model in Section III-A are given as follows. First, the task size and maximum acceptable delay for applications i)-v) are given as: i) [300, 1000] B and 100 ms, ii) [200, 400] B and 120 ms, iii) 300 B and 2000 ms, iv) [300, 1000] B and [1000, 2000] ms (safety) or  $[10^4, 2 \times 10^5]$  ms (route obstruction), and v) [300, 1000] B and [400, 1500] ms, respectively [71]. Furthermore, the computational intensity and the result of the above applications are given as  $[10^3, 10^4]$  cycles/bit and [0.1, 1] KB, respectively [73]. We evaluate the performance of the proposed approach for these vehicular applications in Appendix J of the supplementary material, available online.

### B. The Impact of Multiple Access Schemes

In this sub-section, we discuss the impact of the employed multiple access schemes on the performance. Specifically, in Appendix K of the supplementary material, available online, we evaluate the performance of the proposed BARGAIN-MATCH in the scenario where the network employs the OFDMA, followed by the discussion on the extensibility of the proposed approach for more complicated scenarios.

## VIII. CONCLUSION

In this work, we investigate the computation allocation and task offloading for VEC servers and vehicles in VEC networks. First, to coordinate the space-time-requirement heterogeneity among tasks and the computational heterogeneity among servers, this work employs a hierarchical framework where the intra-server resource allocation and inter-server offloading are decided through the horizontal and vertical collaboration

among vehicle, edge, and cloud layers under the coordination of the controller. Furthermore, JRATOP is formulated to maximize the system utility by jointly optimizing the strategies of resource allocation, resource pricing, and task offloading. To solve the NP-hard problem, we propose the BARGAIN-MATCH that consists of the bargaining-based trading model for intra-server resource allocation and a matching-based collaboration approach for inter-server task offloading. Besides, the proposed BARGAIN-MATCH is proved to be stable, weak Pareto optimal, and polynomial complex. Simulation results demonstrate that BARGAIN-MATCH achieves superior performance in terms of the system utility, vehicle utility and server utility compared to the conventional approaches. Moreover, it can improve the task processing rate and task processing delay significantly, especially when the system workload is heavy.

## REFERENCES

- [1] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for Internet of Things realization," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 2961–2991, Fourth Quarter 2018.
- [2] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, and F. Giust, "Mobile-edge computing architecture: The role of MEC in the Internet of Things," *IEEE Consum. Electron. Mag.*, vol. 5, no. 4, pp. 84–91, Oct. 2016.
- [3] W. Duan, J. Gu, M. Wen, G. Zhang, Y. Ji, and S. Mumtaz, "Emerging technologies for 5G-IoV networks: Applications, trends and opportunities," *IEEE Netw.*, vol. 34, no. 5, pp. 283–289, Sep./Oct. 2020.
- [4] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
- [5] S.-C. Lin, K.-C. Chen, and A. Karimoddini, "SDVEC: Software-defined vehicular edge computing with ultra-low latency," *IEEE Commun. Mag.*, vol. 59, no. 12, pp. 66–72, Dec. 2021.
- [6] L. Dai, B. Wang, Z. Ding, Z. Wang, S. Chen, and L. Hanzo, "A survey of non-orthogonal multiple access for 5G," *IEEE Commun. Surv. Tut.*, vol. 20, no. 3, pp. 2294–2323, Third Quarter 2018.
- [7] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Commun. Surv. Tut.*, vol. 19, no. 3, pp. 1657–1681, Third Quarter 2017.
- [8] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.
- [9] S. Bi, L. Huang, and Y.-J. A. Zhang, "Joint optimization of service caching placement and computation offloading in mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 7, pp. 4947–4963, Jul. 2020.
- [10] H. Yu, Z. Zhou, Z. Jia, X. Zhao, L. Zhang, and X. Wang, "Multi-timescale multi-dimension resource allocation for NOMA-edge computing-based power IoT with massive connectivity," *IEEE Trans. Green Commun. Netw.*, vol. 5, no. 3, pp. 1101–1113, Sep. 2021.
- [11] C. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.
- [12] P. A. Apostolopoulos, E. Tsiropoulou, and S. Papavassiliou, "Risk-aware data offloading in multi-server multi-access edge computing environment," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1405–1418, Jun. 2020.
- [13] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3282–3299, Apr. 2020.
- [14] L. Tan, Z. Kuang, L. Zhao, and A. Liu, "Energy-efficient joint task offloading and resource allocation in OFDMA-based collaborative edge computing," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1960–1972, Mar. 2022.
- [15] Y. Sun et al., "Adaptive learning-based task offloading for vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3061–3074, Apr. 2019.
- [16] 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.
- [17] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.
- [18] P. Qin, Y. Fu, G. Tang, X. Zhao, and S. Geng, "Learning based energy efficient task offloading for vehicular collaborative edge computing," *IEEE Trans. Veh. Technol.*, vol. 71, no. 8, pp. 8398–8413, Aug. 2022.
- [19] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-aware multi-hop task offloading for autonomous driving in vehicular edge computing and networks," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 19, 2022, doi: [10.1109/TITS.2022.3142566](https://doi.org/10.1109/TITS.2022.3142566).
- [20] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 2, pp. 598–611, Feb. 2022.
- [21] L. Liang, H. Ye, and G. Y. Li, "Spectrum sharing in vehicular networks based on multi-agent reinforcement learning," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2282–2292, Oct. 2019.
- [22] X. Zhu, Y. Luo, A. Liu, N. N. Xiong, M. Dong, and S. Zhang, "A deep reinforcement learning-based resource management game in vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2422–2433, Mar. 2022.
- [23] H. Peng and X. Shen, "Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2416–2428, Fourth Quarter 2020.
- [24] W. Duan, X. Gu, M. Wen, Y. Ji, J. Ge, and G. Zhang, "Resource management for intelligent vehicular edge computing networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9797–9808, Jul. 2022.
- [25] S. Choo, J. Kim, and S. Pack, "Optimal task offloading and resource allocation in software-defined vehicular edge computing," in *Proc. Int. Conf. Inf. Commun. Technol. Convergence*, 2018, pp. 251–256.
- [26] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.
- [27] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.
- [28] S. Li, S. Lin, L. Cai, W. Li, and G. Zhu, "Joint resource allocation and computation offloading with time-varying fading channel in vehicular edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3384–3398, Mar. 2020.
- [29] X. Huang, L. He, and W. Zhang, "Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network," in *Proc. IEEE Int. Conf. Edge Comput.*, 2020, pp. 1–8.
- [30] X. Huang, L. He, X. Chen, L. Wang, and F. Li, "Revenue and energy efficiency-driven delay constrained computing task offloading and resource allocation in a vehicular edge computing network: A deep reinforcement learning approach," *IEEE Internet Thing J.*, vol. 9, no. 11, pp. 8852–8868, Jun. 2022.
- [31] C. Xu, M. Wu, Y. Xu, and Y. Fang, "Uplink low-power scheduling for delay-bounded industrial wireless networks based on imperfect power-domain NOMA," *IEEE Syst. J.*, vol. 14, no. 2, pp. 2443–2454, Jun. 2020.
- [32] Q. Luo et al., "An error rate comparison of power domain non-orthogonal multiple access and sparse code multiple access," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 500–511, Mar. 2021.
- [33] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 587–597, Mar. 2018.
- [34] H. Wu et al., "Resource management in space-air-ground integrated vehicular networks: SDN control and AI algorithm design," *IEEE Wireless Commun.*, vol. 27, no. 6, pp. 52–60, Dec. 2020.
- [35] L. P. Qian, Y. Wu, B. Ji, L. Huang, and D. H. K. Tsang, "HybridIoT: Integration of hierarchical multiple access and computation offloading for IoT-based smart cities," *IEEE Netw.*, vol. 33, no. 2, pp. 6–13, Mar./Apr. 2019.
- [36] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus ADMM approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 5087–5099, May 2019.
- [37] A. W. Malik, T. Qayyum, A. U. Rahman, M. A. Khan, O. Khalid, and S. U. Khan, "xFogSim: A distributed fog resource management framework for sustainable IoT services," *IEEE Trans. Sustain. Comput.*, vol. 6, no. 4, pp. 691–702, Fourth Quarter 2021.

- [38] H. Liao et al., "Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5127–5139, Aug. 2021.
- [39] T. Kim et al., "MoDEMS: Optimizing edge computing migrations for user mobility," in *Proc. IEEE Conf. Comput. Commun.*, 2022, pp. 1159–1168.
- [40] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1002–1016, Apr. 2017.
- [41] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.
- [42] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.
- [43] C. Yi, J. Cai, and Z. Su, "A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications," *IEEE Trans. Mobile Comput.*, vol. 19, no. 1, pp. 29–43, Jan. 2020.
- [44] D. Xu, Q. Li, and H. Zhu, "Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 704–707, Apr. 2019.
- [45] F. Fang, K. Wang, Z. Ding, and V. C. M. Leung, "Energy-efficient resource allocation for NOMA-MEC networks with imperfect CSI," *IEEE Trans. Commun.*, vol. 69, no. 5, pp. 3436–3449, May 2021.
- [46] Y. Pan, M. Chen, Z. Yang, N. Huang, and M. Shikh-Bahaei, "Energy-efficient NOMA-based mobile edge computing offloading," *IEEE Commun. Lett.*, vol. 23, no. 2, pp. 310–313, Feb. 2019.
- [47] L. Qian, Y. Wu, F. Jiang, N. Yu, W. Lu, and B. Lin, "NOMA assisted multi-task multi-access mobile edge computing via deep reinforcement learning for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5688–5698, Aug. 2021.
- [48] D. K. Patel et al., "Performance analysis of NOMA in vehicular communications over i.n.i.d Nakagami- $m$  fading channels," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6254–6268, Oct. 2021.
- [49] X. Liu et al., "Placement and power allocation for NOMA-UAV networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 3, pp. 965–968, Jun. 2019.
- [50] Y. Liu et al., "Evolution of NOMA toward next generation multiple access (NGMA) for 6G," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 4, pp. 1037–1071, Apr. 2022.
- [51] Y. Liu, W. Yi, Z. Ding, X. Liu, O. A. Dobre, and N. Al-Dhahir, "Developing NOMA to next generation multiple access: Future vision and research opportunities," *IEEE Trans. Wireless Commun.*, vol. 29, no. 6, pp. 120–127, Dec. 2022.
- [52] Q. Luo et al., "An error rate comparison of power domain non-orthogonal multiple access and sparse code multiple access," *IEEE Open J. Commun. Soc.*, vol. 2, pp. 500–511, Mar. 2021.
- [53] F. Fang, Y. Xu, Z. Ding, C. Shen, M. Peng, and G. K. Karagiannis, "Optimal resource allocation for delay minimization in NOMA-MEC networks," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7867–7881, Dec. 2020.
- [54] B. Makki, K. Chitti, A. Behravan, and M. Alouini, "A survey of NOMA: Current status and open research challenges," *IEEE Open J. Commun. Soc.*, vol. 1, pp. 179–189, Feb. 2020.
- [55] 3GPP, "Technical specification group radio access network; study on non-orthogonal multiple access (NOMA) for NR (Release 16)," 3GPP Sophia Antipolis Valbonne, France, Tech. Rep. TR 38.812, 2018.
- [56] Y. Liu, F. R. Yu, X. Li, H. Ji, and V. C. Leung, "Distributed resource allocation and computation offloading in fog and cloud networks with non-orthogonal multiple access," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 137–12 151, Dec. 2018.
- [57] B. Yang, G. Mao, M. Ding, X. Ge, and X. Tao, "Dense small cell networks: From noise-limited to dense interference-limited," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4262–4277, May 2018.
- [58] A. Boumaalif and O. Zytoune, "Power distribution of device-to-device communications under Nakagami fading channel," *IEEE Trans. Mobile Comput.*, vol. 21, no. 6, pp. 2158–2167, Jun. 2022.
- [59] Y. Pan, C. Pan, K. Wang, H. Zhu, and J. Wang, "Cost minimization for cooperative computation framework in MEC networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 6, pp. 3670–3684, Jun. 2021.
- [60] G. Zhang, W. Zhang, Y. Cao, D. Li, and L. Wang, "Energy-delay tradeoff for dynamic offloading in mobile-edge computing system with energy harvesting devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4642–4655, Oct. 2018.
- [61] Z. Ning et al., "Intelligent edge computing in internet of vehicles: A joint computation offloading and caching solution," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 4, pp. 2212–2225, Apr. 2021.
- [62] A. Rubinstein, "Perfect equilibrium in a bargaining model," *Econometrica: J. Econometric Soc.*, vol. 50, pp. 97–109, 1982.
- [63] D. Gusfield and R. W. Irving, *The Stable Marriage Problem: Structure and Algorithms*. Cambridge, MA, USA: MIT Press, 1989.
- [64] Z. Zhang and R. Q. Hu, "Dense cellular network analysis with LoS/NLoS propagation and bounded path loss model," *IEEE Commun. Lett.*, vol. 22, no. 11, pp. 2386–2389, Nov. 2018.
- [65] C. Liu, K. Li, J. Liang, and K. Li, "COOPER-MATCH: Job offloading with a cooperative game for guaranteeing strict deadlines in MEC," *IEEE Trans. Mobile Comput.*, early access, Jun. 11, 2019, doi: 10.1109/TMC.2019.2921713.
- [66] Y. Hui, Z. Su, T. H. Luan, C. Li, G. Mao, and W. Wu, "A game theoretic scheme for collaborative vehicular task offloading in 5G HetNets," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 16 044–16 056, Dec. 2020.
- [67] A. Bazzi, B. M. Masini, A. Zanella, and I. Thibault, "On the performance of IEEE 802.11 p and LTE-V2V for the cooperative awareness of connected vehicles," *IEEE Trans. Veh. Technol.*, vol. 66, no. 11, pp. 10 419–10 432, Nov. 2017.
- [68] X. Lyu, H. Tian, W. Ni, Y. Zhang, P. Zhang, and R. P. Liu, "Energy-efficient admission of delay-sensitive tasks for mobile edge computing," *IEEE Trans. Commun.*, vol. 66, no. 6, pp. 2603–2616, Jun. 2018.
- [69] H. Guo, J. Liu, and H. Qin, "Collaborative mobile edge computation offloading for IoT over fiber-wireless networks," *IEEE Netw.*, vol. 32, no. 1, pp. 66–71, Jan./Feb. 2018.
- [70] S. A. Kazmi et al., "A novel contract theory-based incentive mechanism for cooperative task-offloading in electrical vehicular networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 8380–8395, Jul. 2022.
- [71] G. A. Association et al., "C-V2X use cases, methodology, examples and service level requirements," White Paper, Jun. 2019.
- [72] M. E. ISG, "Multi-access edge computing (MEC); study on MEC support for V2X use cases," ETSI, Sophia-Antipolis, France, Tech. Rep. GR MEC, vol. 22, 2018.
- [73] F. Spinelli and V. Mancuso, "Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility," *IEEE Commun. Surv. Tut.*, vol. 23, no. 1, pp. 596–630, First Quarter 2021.



**Zemin Sun** (Student Member, IEEE) received the BS degree in software engineering, and the MS and PhD degrees in computer science and technology from Jilin University, Changchun, China, in 2015, 2018, and 2022, respectively. Her research interests include vehicular networks, edge computing, and game theory.



**Geng Sun** (Member, IEEE) received the BS degree in communication engineering from Dalian Polytechnic University, in 2011, and the PhD degree in computer science and technology from Jilin University, in 2018. He was a visiting researcher with the School of Electrical and Computer Engineering, Georgia Institute of Technology, USA. He is an associate professor with the College of Computer Science and Technology, Jilin University, and His research interests include wireless networks, UAV communications, collaborative beamforming, and optimizations.



**Yanheng Liu** received the MSc and PhD degrees in computer science from Jilin University, People's Republic of China. He is currently a professor with Jilin University, People's Republic of China. His primary research interests include network security, network management, mobile computing network theory and applications, etc. He has co-authored more than 90 research publications in peer reviewed journals and international conference proceedings of which one has won "best paper" awards. Prior to joining Jilin University, he was visiting scholar with the University of Hull, England, University of British Columbia, Canada and Alberta University, Canada.



**Jian Wang** (Member, IEEE) received the BSc, MSc, and PhD degrees in computer science from Jilin University, Changchun, China, in 2004, 2007, and 2011, respectively. He is currently a professor with the College of Computer Science and Technology, Jilin University. He is interested in topics related to wireless communication and vehicular networks, especially for network security and privacy protection. He has published more than 40 articles in international journals.



**Dongpu Cao** (Senior Member, IEEE) received the PhD degree from Concordia University, Canada, in 2008. He is a professor with Tsinghua University. His current research focuses on driver cognition, automated driving, and social cognitive autonomous driving. He has contributed more than 200 papers and 3 books. He received the SAE Arch T. Colwell Merit Award in 2012, IEEE VTS 2020 Best Vehicular Electronics Paper Award and more than 10 Best Paper Awards from international conferences. He has served as deputy editor-in-chief of the *IET Intelligent Transport Systems Journal*, and an associate editor of *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE/ASME Transactions on Mechatronics*, *IEEE Transactions on Industrial Electronics*, *IEEE/CAA Journal of Automatica Sinica*, *IEEE Transactions on Computational Social Systems*, and *ASME Journal of Dynamic Systems, Measurement and Control*. He is an IEEE VTS distinguished lecturer.