# Task Offloading and Resource Allocation for IoV using 5G NR-V2X Communication

Salman Raza, Shangguang Wang, *Senior Member, IEEE,* Manzoor Ahmed, Muhammad Rizwan Anwar, Muhammad Ayzed Mirza, and Wali Ullah Khan, *Member, IEEE*

***Abstract*—Vehicular edge computing (VEC) is an innovative computing paradigm with an exceptional ability to improve the vehicles' capacity to manage computation-intensive applications with both low latency and energy consumption. Vehicles require to make task offloading decisions in dynamic network conditions to obtain maximum computation efficiency. In this paper, we analyze computation efficiency in a VEC scenario, where a vehicle offloads its tasks to maximize computation efficiency as a tradeoff between computation time and energy consumption. Although, it is quite a challenge to ensure the quality of experience of the vehicle due to diverse task requirements and the dynamic wireless conditions caused by vehicle mobility. To tackle this problem, a computation efficiency problem is formulated by jointly optimizing task offloading decision and computation resource allocation. We propose a Mobility-Aware Computational Efficiency based Task Offloading and Resource Allocation (MACTER) scheme and develop a distributed MACTER algorithm that provides the best solution. We further consider the fifth-generation new-radio vehicle-to-everything communication model, i.e., cellular link and millimeter wave, to enhance the system performance. The simulation outcomes demonstrate that the proposed algorithm can efficiently enhance computation efficiency while satisfying computing time and energy consumption constraints.**

***Index Terms*—Computation resource allocation, mobility, task offloading, Vehicular Edge Computing**

## I. Introduction

WITH the progression in the Internet-of-Vehicles (IoV) and wireless technologies, smart vehicles, i.e., autonomous vehicles, are becoming increasingly popular, which has led to new applications with advanced features. Autonomous vehicles need to extract meaningful information from the massive amount of data collected by sensors, which comprehend the environment, and make decisions depending on the constantly occurring changes [1]. While, cutting-edge technologies require high-performance computation and stringent real-time response, i.e., vision-based object detection, self-driving, and immersive gaming [2], [3]. Nevertheless, these applications are usually computation-intensive, energy-consuming, and delay-sensitive. It is tough to manage the computation demands of such complex applications due to the limited computation capability of vehicles. Thus, it results in bottlenecks and makes it extremely difficult for the vehicles to meet their resource requirements and guarantee the required quality of experience (QoE) level [4].

Cloud computing infrastructure is in place for many years to deal with resource-intensive applications. However, cloud resources are placed far away from users, causing excessive and expensive bandwidth issues, not supporting delay-sensitive applications, and security and privacy problems [5]. Consequently, it is a requisite to bring the resources close to the network edge to fully support dynamic scalability, network processing efficiency, and modify computing paradigms design [6]. In this regard, Multi-access Edge Computing (MEC) complemented cloud computing and enabled users to reduce latency and save energy by offloading computation towards the edge servers [7]. However, for high mobility scenarios due to a short validity period of high-speed mobility, the conventional MEC-based offloading approach is incompetent in the vehicular environment [8].

Nowadays, much work is being done to merge MEC technology into a vehicular network in academics and industry. Specifically, Vehicle edge computing (VEC) is the MEC technology associated with the vehicular network. VEC is extremely useful for carrying out computation-intensive and time-constrained tasks under vehicular networks [8], [9]. Through offloading complex computational tasks over VEC servers, computing delay and energy consumption of vehicular applications can be drastically minimized while mitigating the chance of network congestion. In addition, sometimes it is not feasible to offload tasks to edge servers as it uses extra energy and consumes more time [10]. The challenge, however, is to make the offloading decision while taking overall computation and communication costs into account.

On the other hand, vehicles face certain unprecedented constraints, although they are capable of executing more computational tasks. These constraints include inadequate computing capacity and high energy consumption [11]. Owing to these constraints, some of the typical challenging scenarios are:

- As vehicles have limited computation and energy resources, how to meet vehicles' real-time stringent time and energy demands?
- In the case of autonomous vehicles, where many computation-intensive applications require a huge amount

Salman Raza and Muhammad Rizwan Anwar are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China e-mail: (salmanraza@bupt.edu.cn; m.r.anwar@bupt.edu.cn).

Shangguang Wang is with Beiyou Shenzhen Research Institute, Shenzhen, China. (e-mail:sgwang@bupt.edu.cn). Shangguang Wang is the corresponding author

Manzoor Ahmed is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China. (e-mail:manzoor.achakzai@gmail.com)

Muhammad Ayzed Mirza is with the School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China. (e-mail:mamirza@bupt.edu.cn)

Wali Ullah Khan is with the Interdisciplinary Center for Security, Reliability and Trust (SnT), University of Luxembourg, 1855 Luxembourg City, Luxembourg. (e-mail:waliullah.khan@uni.lu, waliullahkhan30@gmail.com)

of energy usage, how can vehicles guarantee their mileage durability?
- Coupled with the above points, that autonomous vehicles generate a massive amount of data from their environment at each moment, how can it be managed, transmitted, and stored effectively?
- As the vehicular computing capability cannot cope with the ever-increasing computing demands, how can the high cost of hardware upgrades be borne?

All of the above-mentioned challenges are related to time constraints and energy consumption. The task offloading approach offers feasible solutions to tackle the aforementioned problems.

### A. Related Work

VEC and task offloading have been a trending topic in recent years. Many researchers have done a great deal of significant work in this domain [12]–[16]. For instance, in [12], Gu *et al.* presented a distributed context-aware task offloading approach based on matching theory. The authors minimized the system delay by developing two heuristic algorithms. Wu *et al.* [13] proposed an edge-cloud collaborative reinforcement learning based scheme to find optimal routes with a low communication overhead. Zhou *et al.* [14] introduced a shared dedicated short-range communication and fifth-generation (5G) communication framework to support immersive experience vehicular network. Dai *et al.* [15] analyzed the integration of offloading with load balancing and considered resource allocation for a multi-user multi-server vehicular network. Zhou *et al.* [16] examined the offloading problem by presenting a distributed low-complexity approach to identify offloading the optimum part of the task, which depends on the latency and energy consumption in the vehicle computing, task transmission, task processing, and task handover. These studies are primarily concerned with high-reliability and low-latency networks and did not address the energy-saving issues, particularly with limited battery Electrical Vehicles (EVs).

Many studies in edge computing examine the energy efficiency problem by considering task offloading and resource allocation. Deng *et al.* [17] optimized the resource allocation between cloud and fog in order to minimize the energy utilization with various latency constraints. Yuan *et al.* [18], focused on the problem of profit optimization for edge-cloud service providers. The authors formulated this problem by considering the constraint of maximum response time, and the revenue and the penalty cost for each task are determined through service-level agreements. In [19] proposed a task offloading scheme to minimize the total energy consumed by both mobile devices and edge servers. The authors solved this problem by a hybrid meta-heuristic algorithm to produce a near-optimal solution. Chen *et al.* [20] formulated the stochastic task offloading problem in a sliced radio access network as a Markov decision process to maximize the long-term utility performance by considering the time-varying communication qualities and computation resources. Nevertheless, the aforementioned work primarily targets static mobile networks and can not be used directly for extremely dynamic vehicle networks. Although some of the works applied MEC for vehicular networks [9], [21], [22], primarily focused on the offloading issue from a perspective of latency minimization and did not analyze the vehicular energy efficiency concerns, especially with limited battery capacity. On the other hand, some studies [23]–[26] focused on performing computational offloading to reduce vehicle energy consumption. In addition, the majority of the prior solutions depend on centralized methods of optimization, where the computational complexity increases dramatically with the number of vehicles. It is much convenient to deal with the issue from a distributed aspect, considering complexity and scalability issues. There is also a lack of a unified distributed approach to tackle vehicle latency and energy-saving issues related to vehicle mobility considerations.

In addition, the game theory can answer the decision-making problem between the multiple players to achieve the goal. The task offloading studies relying on game theory was considered in [27]–[31]. In [27], the authors introduced a game-theoretic analysis for multi-user and developed a distributed task offloading algorithm that achieves a Nash equilibrium (NE). Liu *et al.* [28] considered the task offloading problem to minimize communication burden on the edge server. They used game theory to select suitable channels and make the optimal offloading decisions. In [29], a multilevel offloading approach according to the Stackelberg game theory was designed, which maximizes both vehicle and server revenues. The authors in [30] presented a Bayesian coalition game to enhance the computing resource utilization and minimize energy consumption in a vehicular cloud. In [31], Huang *et al.* analyzed a task offloading problem in which parked vehicles act as servers and use blockchain to offload computation in a decentralized manner. The authors defined and solved this problem by utilizing the Stackelberg game framework in order to minimize total payments for users. In [32], Zhan *et al.* presented a task offloading approach by combining proximal policy optimization and convolutional neural networks. The authors considered tasks without stringent latency requirements or execution priority.

However, some prior studies analyzed to optimize the task offloading or computation resource allocation strategies without simultaneously optimizing them. Similar to the studies presented in [27], [33], the authors only analyzed task offloading but did not incorporate the computation resource allocation since each vehicle generally receives various computational tasks in real-time that need different computational resources. Besides, the tasks in [34] and [35] ignored the optimization of task offloading, in which entire tasks were offloaded to the MEC server. In task offloading, vehicles share limited computing and communication resources and each vehicle must determine where its task will be processed and whether it is appropriate to offload. Therefore, to improve system performance, the task offloading and resource allocation approach must be optimized. Moreover, most of the studies on vehicular task offloading did not examine the task offloading and resource allocation with both stringent latency demands and energy requirements and ignored this significant factor. Unlike the preceding task offloading techniques for VEC, we develop a multi-vehicle task offloading game considering the task deadline and energy consumption constraints while keeping in view the vehicle mobility. We propose a distributed mobility-aware computational efficiency based task offloading and resource allocation algorithm, and guarantee that decision of each vehicle will converge to the NE.

### B. Contributions

In this article, we analyze the computation efficiency problem for EVs, i.e., autonomous vehicles, where an EV traveling along an urban road decides where to offload its task, to enhance the computation efficiency. The tasks are independently generated by various applications with different characteristics such as task data size, required CPU cycle, and energy consumed. Besides, the vehicles' mobility causes the transmission rate to vary periodically over time. These factors result in dynamically changing the vehicular task computation time and energy consumption. Owing to the highly dynamic scenario, designing an efficient offloading approach is quite challenging. As a performance metric, we introduce the computation efficiency, which is the ratio of computed bits to the total energy consumed, it can achieve time and energy consumption minimization. The key contributions of this article are listed as follows:

1) This paper aims to improve computation efficiency by optimizing the task offloading and resource allocation in compliance with EV's time and energy constraints. The computation efficiency is the ratio of computed bits to energy consumption and can lead to an efficient vehicles' time and energy utilization.
2) A Mobility-Aware Computational Efficiency based Task Offloading and Resource Allocation (MACTER) scheme is proposed to make optimal decisions. A game-theoretical method is adopted for making the offloading decision. While the resource allocation is performed by the Lagrange multiplier technique. In addition, we use 5G new-radio vehicle-to-everything (NR-V2X) based millimeter wave (mmWave) technology which improves overall system performance.
3) A distributed MACTER algorithm is designed for our scheme. It runs within the offloading strategies and resource allocation iteratively to achieve NE. We perform extensive simulations to check and validate the proposed approach. The experimental findings reveal that by comparing with benchmark approaches, the proposed algorithm effectively improves the computation efficiency while satisfying both the task computation time and energy consumption.

### C. Paper Organization

The remaining parts of this article are structured as follows. In Section II, we present the system model, and the problem formulation is discussed in Section III. The mobility-aware computational efficiency based task offloading and resource allocation scheme and a distributed algorithm are provided in Section IV. The numerical results are provided in Section V. Finally, Section VI concludes this article.

## II. System Model

In this section, we introduce the network topology and the communication model, followed by the computation model. Then, the vehicle and VEC utility functions are discussed in detail. Table I represents all the notations to be used in this section.

TABLE I: Frequently Used Notations

| Symbols | Description |
|---|---|
| $G_b^i(\phi)$ | Function of steering angle |
| $\gamma$ | Maximum rate of RSU |
| $\gamma_i$ | Minimum between $\gamma$ & $\gamma_i^{'}$. |
| $\gamma_i^{'}$ | Data uplink rate of vehicle $i$. |
| $r$ | Radius of the RSU |
| $C_i$ | Computation resource needed to complete a task |
| $\alpha_i^{in}$ | Task data size |
| $t_i^{max}$ | Maximum tolerable delay |
| $t_i^{stay}$ | Vehicle's stay time in RSU |
| $t_i^{ptd}$ | $\min\{t_i^{max}, t_i^{stay}\}$, means, it is practical tolerable delay of a task |
| $t_i^{loc}$ | Local computing time |
| $t_i^{vec}$ | VEC offloading time i.e., Transmission + Processing |
| $e_i^{loc}$ | Local energy consumption |
| $e_i^{vec}$ | Vehicle energy consumption while offloading to VEC |
| $E_i$ | Entire energy consumption |

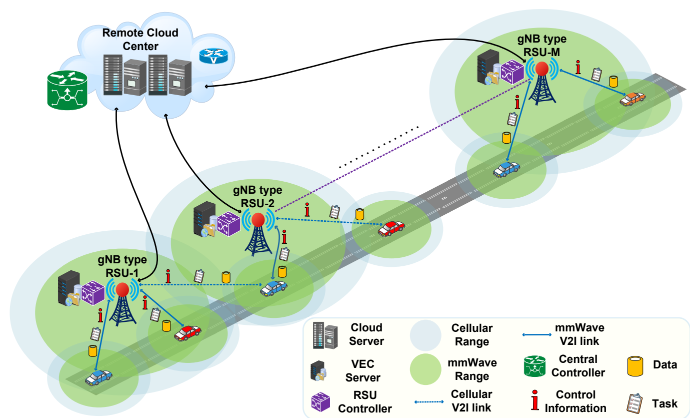



Fig. 1: The Task offloading under both cellular and mmWave technologies in VEC Computing Network.

### A. Network Topology

Fig. 1 illustrates our proposed VEC network, which comprises of $N$ number of vehicles. Here, a unidirectional road is considered in an urban area, where $M$ roadside units (RSUs) with equal communication range $r$ are located along the road, each having a VEC server consisting of $F$ computing resource. Where the id set of RSUs is represented as $\mathcal{M} = \{1, 2, 3, ..., M\}$. Accordingly, we can distribute the road segment into $M$ segments, and each vehicle is randomly distributed in the urban area. We also consider the vertical distance between the road and RSU, which is denoted by $e$. The vehicles' set is represented as $\mathcal{N} = \{1, 2, 3, ..., N\}$. Each vehicle has a computation task, either computed locally or offloaded to the VEC server. The tasks are offloaded to

VEC servers via RSU. The vehicles can connect to RSU $m$, i.e., $m \in \mathcal{M}$, when they move within the $mth$ segment. In addition, Fig. 2 represents the architecture of the VEC computing network.

We then define vehicular offloading strategy, where $\mathcal{D} = \{d_i | d_i \in \{d_i^{loc}, d_i^{vec}\}, d_i^j \in \{0,1\}, i \in \mathcal{N}, j \in \{loc, vec\}\}$ as the vehicular offloading decision. The offloading decision set is denoted as $\mathcal{A} = \{loc, vec\}$, which represents the decision for local or VEC offloading, respectively. Moreover, $d_i = d_i^j = 1$ implied the completion of the vehicle $i$ task by choosing the decision $j$, on the contrary $d_i = d_i^j = 0$.

*1) Mobility Model:* We assume that each vehicle enters the road segment at a random speed. The vehicle speeds are independent and identically distributed. Generally, the speed of each vehicle can vary with time as it moves along the road. Therefore, each vehicle is assigned a random speed $\upsilon$ chosen from the Gaussian distribution, and each vehicle maintains its given speed. In order to avoid the negative speed of vehicles, a truncated Gaussian distribution is used. Further, we apply a truncated Gaussian probability density function (PDF), which is defined as:

$$f_\upsilon^\sim = \frac{2 f_\upsilon(\upsilon)}{erf(\frac{V_{max}-\mu}{\vartheta\sqrt{2}}) - erf(\frac{V_{min}-\mu}{\vartheta\sqrt{2}})}, \tag{1}$$

where $f_\upsilon(\upsilon) = \frac{1}{\vartheta\sqrt{2}} exp(-\frac{(\upsilon-\mu)^2}{2\vartheta^2})$ is the Gaussian PDF, $V_{max} = \mu + 3\vartheta$ is the maximum velocity, and $V_{min} = \mu - 3\vartheta$ is minimum vehicular velocity, $erf(.)$ is error function, $\mu$ is the average speed, and $\vartheta$ is defined as a standard deviation of vehicular speed [36]. Thus, according to (1), a corresponding speed $(V_{min} \le \mu_\upsilon \le V_{max})$ is given by

$$\mu_\upsilon = \frac{1}{\int_{V_{min}}^{V_{max}} \frac{f_\upsilon^\sim(\upsilon)}{\upsilon} d\upsilon} = \frac{erf(\frac{V_{max}-\mu}{\vartheta\sqrt{2}}) - erf(\frac{V_{min}-\mu}{\vartheta\sqrt{2}})}{\frac{2}{\vartheta\sqrt{2\pi}} \int_{V_{min}}^{V_{max}} \frac{exp(-\frac{(\upsilon-\mu)^2}{2\vartheta^2})}{\upsilon} d\upsilon} \tag{2}$$

where the values for $\mu$ and $\vartheta$ are derived according to the measurements formulated in [37].

Moreover, while offloading the task, it must be ensured that the task is completed before the vehicles switch from their corresponding RSU to another RSU and the computation task delay is satisfied. For this, we need to find the stay time of a vehicle, which is described as follows:

*2) Vehicle's Stay Time:* Since the RSU communication radius $r$, and the vertical distance $e$ between the RSU and road are defined. Besides, the $\upsilon_i$ is the velocity of a vehicle. Thus, we can derive the stay time of the vehicle $i$ as:

$$t_i^{stay} = \frac{2\sqrt{r^2 - e^2}}{\upsilon_i} \tag{3}$$

We define the stay time as the time vehicle $i$ stays within the communication range of its corresponding RSU $m$.

### *B. Communication Model*

In the vehicle to infrastructure (V2I) communication, we consider that the vehicles interact with the RSUs according to Mode-1's cellular links of 5G NR-V2X using cellular and mmWave communication links. We consider that each vehicle and RSU have mmWave as well as cellular network facilities, which are both installed with multiple antennas enabling communication over mmWave and 5G links. The communication models are dependent upon a certain distance between the vehicle and RSU. Since the mmWave-based V2X could achieve an ultra-high rate of up to 7 Gbps using mmWave within a range of 300 [38]. Moreover, according to [39], the distance between two network components is recommended from 100 m to 200 m. Therefore, we assume the mmWave communication range to 150 m and cellular link range to 200 m. The following subsection describes both cellular and mmWave links in detail.

*1) Cellular mode:* In V2I communication, the cellular link lies under the Mode-1 of NR-V2X [40]. 5G NR-V2X is developed in 3GPP Rel. 16, introducing the first V2X standard, based on 5G NR that was standardized in 3GPP Rel. 15. The NR V2X is capable of supporting advanced V2X applications with more stringent QoS requirements than those supported by Cellular V2X [41]. The 5G NR-V2X in the 3GPP context ensures improved performance in terms of throughput, latency, reliability, connectivity, and mobility [42]. The implementation of the gNBs in this paper is considered as standalone node connected to pure 5G system core and access components, which are collocated with user plane function (UPF) and V2X application server. While the RSUs are assumed to be implemented as standalone UE-type RSU node following 3GPP Rel.16 NR-V2X standards. The data transmission rate between vehicle $i$ and the RSU $m$ is derived as:

$$\gamma_i^{'} = W_{uu} \log_2 \left(1 + \frac{p_{i,m}[r\lceil \frac{s_i}{r} \rceil - s_i]^{-\delta} |h|^2}{\sigma_{uu}^2}\right), \tag{4}$$

where $W_{uu}$ is the channel bandwidth, $p_{i,m}$ is vehicle $i$'s transmission power over its corresponding RSU. The distance traveled by vehicle is $[r\lceil \frac{s_i}{r} \rceil - s_i]$, where $s_i$ denotes the vehicle $i$ current position and the factor $\delta$ is the path loss exponent [43]. In addition, the uplink channel is modeled as the Rayleigh fading channel defined as $|h|^2$ [44], and $\sigma_{uu}^2$ is the Gaussian noise.

*2) mmWave mode:* To exploit the benefits of utilizing mmWave in V2I communication mode, each vehicle and RSU are assumed to be installed with directional antennas, and the vehicle $i$ antenna gain is modeled as a function of steering angle is given as $\phi$. Furthermore, the antenna gain $\mathcal{G}_b^i(\phi)$ of a generic mmWave is expressed as:

$$\mathcal{G}_b^i(\phi) = \begin{cases} \mathcal{G}_i^{max}, & if\ |\phi| \le \phi_b \\ \mathcal{G}_i^{min}, & otherwise\ , \end{cases} \tag{5}$$

where $\phi$ is the angle off to boresight direction, $\mathcal{G}_i^{max}$ and $\mathcal{G}_i^{min}$are the array gains of main lobe and side lobe, respectively, and $\phi_b$ is the main lob's beam-width. Moreover, the transmission rate of vehicle $i \in \mathcal{N}$ and RSU $m \in \mathcal{M}$ is calculated in $\gamma_i^{'}$ as represented in (6), where $W_{mm}$ is the mmWave channel bandwidth and can be expressed as:

$$\gamma_i^{'} = W_{mm} \log_2(1 + SNR_{i,m}), \tag{6}$$

where, $SNR_{i,m}$ is the SNR between vehicles and the associated RSU in mmWave mode is defined as:

$$SNR_{i,m} = (p_{i,m} - \sigma_{mm} - 10log_{10}(W_{mm}) + \mathcal{G}_i^{max}\mathcal{G}_m^{max} - 10\zeta log_{10}([r\lceil\frac{s_i}{r}\rceil - s_i]) - 69.6 - \rho\alpha, \tag{7}$$

where the $\zeta$ is the path loss exponent and $\rho_\alpha$ is the shadow fading set to 3 dB in line of sight scenarios [45].

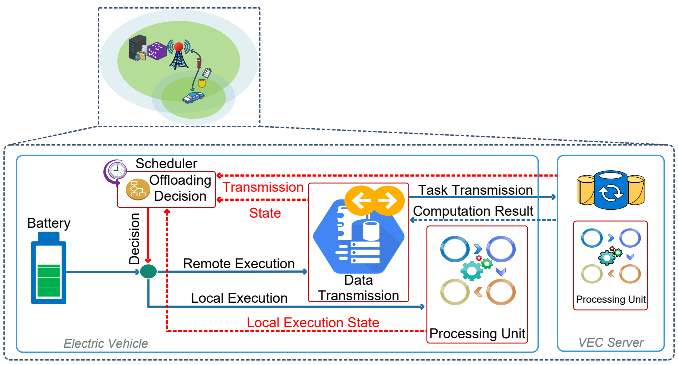



Fig. 2: Architecture for VEC Computing Network.

### C. Computation Model

We assume that each vehicle contains a computation task $\varphi_i = \{C_i, \alpha_i^{in}, t_i^{max}\}$, where $C_i = \hbar_i\alpha_i^{in}$, $\alpha_i^{in}$ represents the data size of the task , $C_i$ is required computation resource to complete the task $\varphi i, \hbar_i$ is the service coefficient that defines the relationship of $C_i$ and $\alpha_i^{in}$, and $t_i^{max}$ is the task maximum tolerable delay. Therefore, by considering practical assumption, we can derive from (3) that the practical tolerable delay of the task must be $t_i^{ptd} = \min\{t_i^{max}, t_i^{stay}\}$. This ensures that while satisfying the task delay constraint, the vehicle remains within the range of connected RSU. We present local computing and VEC computing to further illustrate the computation model.

*1) Local Computing:* When vehicle $i$ computes the task locally, the computing time and energy consumption rely on its available resources. It is considered that $f_i^{loc}$ is the vehicle $i$'s computing resource, and it is defined by the vehicles' onboard unit capacity. Then, the local computing time $t_i^{loc}$ and energy consumption $e_i^{loc}$ can be obtained by (8) and (9), respectively.

$$t_i^{loc} = \frac{C_i}{f_i^{loc}}, \tag{8}$$

$$e_i^{loc} = \varsigma_i C_i, \tag{9}$$

where $\varsigma_i$ in (9) denotes the consumption of energy per computing unit [23].

*2) VEC Computing:* When the local computing is not feasible, then the task is offloaded to the VEC server. Like many prior studies [29], [34], [46], various applications, i.e., speech recognition, the receiving time for computation output is ignored as the output size is substantially smaller than that of the input. In this context, $t_i^{vec}$ shows the VEC execution time and uplink transmission time of vehicle $i$, which can be written as:

$$t_i^{vec} = \frac{C_i}{f_i^{vec}} + \frac{\alpha_i^{in}}{\gamma_i}, \tag{10}$$

where $f_i^{vec}$ represents the assigned computation resource to vehicle $i$ by VEC and $\gamma_i$ is vehicle $i$ accessible data transmission rate. In addition, and $\gamma_i = \min\{\gamma, \gamma_i^{'}\}$, where $\gamma$ is the RSU's maximum data rate, and $\gamma_i^{'}$ is the vehicle's $i$ uplink data rate. The vehicle $i$ energy consumption for transferring the task to the VEC server is expressed as:

$$e_i^{vec} = p_i \times \frac{\alpha_i^{in}}{\gamma_i} \tag{11}$$

where, $p_i$ in (11) shows the average transmission power of vehicle $i$ while offloading. Moreover, the entire energy consumption of the system to execute task $i$ is calculated as:

$$E_i = e_i^{loc} + e_i^{vec} \tag{12}$$

In this work, we assume the computation capacity of the VEC server is adequate, and each offloading vehicle can be allocated $f_i^{vec}$ computation resource.

**Definition 1** *Energy-Time Cost (ETC) is defined as the weighted sum of energy consumption and task executing time.* Therefore, the ETC for vehicle $i$ in local computing is given by

$$K_i^{loc} = \Lambda_i^E e_i^{loc} + \Lambda_i^T t_i^{loc} \tag{13}$$

where $\Lambda_i^E + \Lambda_i^T = 1$, $0 \leq \Lambda_i^E \leq 1$ and $0 \leq \Lambda_i^T \leq 1$ indicate the weights of energy consumption and task executing time for vehicle $i$. To fulfill the specific users' requirements, vehicles are allowed to select different weights to make their decisions. For instance, vehicles with energy priorities would choose a larger $\Lambda_i^E$ to save more energy. Meanwhile, when the vehicle is executing some delay-sensitive applications, e.g., object detection, then it is preferable to fix a larger $\Lambda_i^T$ for delay minimization.

Similarly, (14) is represented as the weighted sum of energy consumption and task execution time. Therefore, the ETC for vehicle $i$ in VEC computing is defined as:

$$K_i^{vec} = \Lambda_i^E e_i^{vec} + \Lambda_i^T t_i^{vec} \tag{14}$$

### D. Utility Functions

In this section, we define a utility function for both vehicle and VEC offloading. It guarantees the level of satisfaction that is taken by the vehicle to take offloading decisions. For the utility function design, we take into account the following metrics.

- Energy Consumption: Energy consumption is a critical metric for EVs. Since it is one of the most concerning issues nowadays. Generally, the utility function is expected to decrease monotonically with increased energy consumption. Moreover, the vehicles' satisfaction should be greater than zero regarding energy consumption.
- Computing Delay: The task computing delay is also a significant metric in making a real-time decision. The vehicle obtains higher satisfaction as the computing delay

is shortened. Similar to the energy consumption metric, the computing delay should also be non-negative.

- Computation Resource Cost: Each vehicle that offloads its tasks to a VEC server must pay for utilizing the VEC servers' resources [29], [46]. Besides, a vehicle that utilizes more computing resources would generate higher costs.

In addition to VEC computing, a vehicle can also use its computing resources to handle computing tasks locally without the need for service payment. However, when the computation delay and energy consumption of local computing are greater than the task's maximum tolerable delay and maximum energy consumption, likewise, the processing parameters of the VEC computing do not exceed the maximum tolerable delay, as well as the vehicles' maximum energy consumption while offloading. Therefore, it is essential to guarantee that the vehicles' utility should be within the limits of VEC computing.

*1) Vehicle Utility Function:* The local computing utility function for vehicle $i$ is represented as:

$$\begin{aligned} U_i^{loc} =& \ln\left(1 + ((\Lambda_i^E e_i^{loc(max)} + \Lambda_i^T t_i^{max}) - K_i^{loc})^+\right) \\ & - \varpi I((\Lambda_i^E e_i^{loc(max)} + \Lambda_i^T t_i^{max}) < K_i^{loc}), \end{aligned} \tag{15}$$

where $I(x)$ is represented as an indicator function, if $x$ is true, it is equal to $1$, otherwise, it is $0$, and $\varpi$ is used to normalize the $U_i^{loc}$. $\varpi$ ensures $U_i^{loc} < d_i^{vec} U_i^{vec}$ when $(\Lambda_i^E e_i^{loc(max)} + \Lambda_i^T t_i^{max}) < K_i^{loc}$ and $d_i^j K_i^{vec} < d_i^{vec}(\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd})$. Where $e_i^{loc(max)}$ is the task's maximum energy consumption. Its value is determined by the highest value of the $\varsigma_i$ energy consumption per computing unit.

*2) VEC Utility Function:* The VEC computing utility function for vehicle $i$ is represented as:

$$\begin{aligned} U_i^{vec} =& \theta_i \ln\left(1 + ((\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) - K_i^{vec})^+\right) \\ & - (1-\theta_i)\rho_{vec} f_i^{vec}, \end{aligned} \tag{16}$$

where $(x)^+$ is the $\max(x, 0)$ that ensures the satisfaction to be always greater than zero concerning the processing delay, the coefficient of weight is represented as $\theta_i$, $\rho_{vec}$ is the computing resource unit cost of VEC server [46]. Whereas $e_i^{vec(max)}$ is the maximum energy consumption while offloading a task. Its value is highly dependent on the transmission rate of vehicle $i$.

Now, the total network computation efficiency ($\mathbb{E}$) can be represented as:

$$\mathbb{E} = \sum_{i=1}^{N} \sum_{j=1}^{J} \frac{\varphi_i d_i^j U_i^j}{E_i}. \tag{17}$$

## III. Problem Formulation

Our main objective is to enhance the computation efficiency of the VEC network as a whole. The computation efficiency is expressed as the ratio of the total computed bits to the energy consumption of the EVs. For this, we formulate an optimization problem to maximize the utility of the system through optimizing task offloading strategy $\mathcal{D}$ and resource allocation $\mathcal{F}$ to eventually improve the overall computation efficiency and is mathematically expressed as:

$$\begin{aligned} \mathbf{P1}: \quad & \max_{\{D\},\{F\}} \ \mathbb{E} \\ s.t. \quad & C1: f_i^{loc} \geq 0 \quad \forall i \in \mathcal{N}, \\ & C2: 0 \leq f_i^{vec} \leq d_i^{vec} F_i^{vec}, \quad \forall i \in \mathcal{N}, \\ & C3: \sum_{i=1}^{N} f_i^{vec} \leq F_i^{vec}, \quad i \in \mathcal{N}, \\ & C4: d_i^{loc} + d_i^{vec} \leq 1, \quad \forall i \in \mathcal{N}, \\ & C5: d_i^j = \{0,1\}, i \in \mathcal{N}, j \in \{loc, vec\} \\ & C6: d_i^j K_i^j \leq d_i^{vec}(\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) \\ & \ + d_i^{loc}(\Lambda_i^E e_i^{loc(max)} + \Lambda_i^T t_i^{max}), \end{aligned} \tag{18}$$

where the constraint C1 is the local computing available resources that are greater or equal to zero, C2 is used to assign the available computational resource to the vehicle $i$ on its corresponding VEC server, and C3 represents the total VEC computation resource. For each task, one approach can be selected that is illustrated in C4 and C5. Also, the latency and energy constraints of each computation task are represented in C6. Moreover, $\mathcal{D} = \{D^{loc}, D^{vec}\}$ is the execution indicator vector and $\mathcal{F} = \{f_1^{vec}, f_2^{vec}, ..., f_N^{vec}\}$ is the resource allocation.

The optimization (18) is non-convex and is a mixed-integer programming problem since the objective function involves sum-of-ratio maximization. To solve it efficiently, a low complexity MACTER scheme is proposed to analyze the computation efficiency. As a result, the original problem (18) is decoupled into two subproblems to tackle both task offloading decisions and computation resource allocation. The game theory approach is adopted to make task offloading decisions. Whereas Lagrange multiplier technique and bisection method are used to solve the computation resource allocation problem, which further ensures the near-optimal resource allocation solution. The game updates the task offloading strategies after the resource allocation stage until NE is achieved. Once all vehicles take the offloading decisions, the computation resource allocation of VEC computing needs optimization to maximize all the offloading vehicles' utilities. Through mutual iteration, the system enters into a steady state and achieves the near-optimal solution. Therefore, decoupling the computational efficiency problem enables us to handle the problem, which will ultimately attain a near-optimal solution like our original problem defined in (18).

## IV. Mobility-Aware Computational Efficiency based Task offloading and Resource Allocation (MACTER) Scheme

In this section, we propose a MACTER scheme to maximize computation efficiency. The computation efficiency depends upon the optimal task offloading and computation resource allocation. Therefore, under the achieved computing resources allocation, the offloading strategy is obtained in this scheme, and resource allocation is observed in the context of a provided offloading strategy. Besides, to maximize computation efficiency, the utility functions are the key attributes to be considered in decision making. Moreover, the detail of the scheme is as follow:

### A. Task offloading

In the task offloading, the vehicles' offloading decision depends upon its offloading demands as well as the offloading strategies of other vehicles. The game theory is viable to resolve the problem regarding decision-making as it provides a robust model to overcome the clash of interests among the vehicles in real-time to make the best decision.

*1) A Game Approach:* The multi-vehicle computing problem is formulated as a task offloading strategy game. All vehicles have computation tasks and are considered players in this game, while players compete for resources in order to maximize their utility. The task offloading strategy game is described as $\Gamma = \{\mathcal{N}, (\mathcal{D}_i)_{i\in\mathcal{N}}, (U_i)_{i\in\mathcal{N}}\}$, where vehicle set is $\mathcal{N}$ and the set of offloading strategy of vehicle $i$ is denoted by $\mathcal{D}_i$. For vehicle $i$, the utility function is given as $U(d_i, d_{-i})$, where $d_{-i} = (d_1, ..., d_{i-1}, d_{i+1}, ..., d_N)$ denotes the other vehicles' offloading strategies except vehicle $i$. The objective of each vehicle is to choose a valuable offloading strategy $(d_i^{vec} + d_i^{loc} = 1)$ to optimize its utility such as:

$$\max_{d_i} U(d_i, d_{-i}) = d_i^{vec} U_i^{vec} + d_i^{loc} U_i^{loc}. \tag{19}$$

Then, a NE is presented to fix the problem of the task offloading strategy game.

***Definition 1:*** *A strategy $\mathcal{D}^* = (\mathcal{D}^{vec*}, \mathcal{D}^{loc*})$ is the NE, if we have the following relationship for any vehicle $i \in N$.*

$$U(d_i^*, d_{-i}^*) \geq (d_i, d_{-i}^*). \tag{20}$$

The NE is achieved when no vehicle intends to unilaterally break the NE steady-state to earn the extra benefit. For the offloading strategy, the existence of NE is verified by introducing an exact potential game. There is always a NE in every potential game given the finite strategy sets and having the finite improvement property (FIP). Furthermore, for each vehicle the exact potential game accepts a potential function $\psi(d)$ while the offloading strategy unilaterally changes from $d_i$ to $d_i^{'}$ and $d_{-i} \in \Pi_{j\neq i}\mathcal{A}_j, d_i, d_i^{'} \in \mathcal{D}_i$. The following relationship can be derived as:

$$U(d_i, d_{-i}) - U(d_i^{'}, d_{-i}) = \psi(d_i, d_{-i}) - \psi(d_i^{'}, d_{-i}) \tag{21}$$

The potential function of a player $\psi(d)$ reveals exactly the unilateral modification performed by the utility function [27].

*Lemma 1:* The task offloading strategy in (22) always converges to the NE with a function $\psi_s$ of exact potential game.

$$\begin{aligned}
&\psi(d) = \\
&d_i^{loc} \sum_{n=1}^{N} (\ln(1 + (\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) - K_n^{loc})^+) \\
&- \varpi I((\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) < K_n^{loc})) + (1 - d_i^{loc}) \\
&\times \Big\{\theta_i \ln(1 + ((\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) - d_i^{vec} K_i^{vec})^+) \\
&- (1-\theta_i)(d_i^{vec} \rho_{vec} f_i^{vec}) \\
&+ \sum_{n=1, n\neq i}^{N} \Big(\ln(1 + ((\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) - K_n^{loc}))^+\Big) \\
&- \varpi I((\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) < K_n^{loc}))\Big\}.
\end{aligned} \tag{22}$$

At the time vehicle $i$ offloads a task to the VEC server or compute it locally, according to (22), it can be observed as:

$$\begin{aligned}
&\psi(d_i^{vec}, d_{-i}) - \psi(d_i^{loc}, d_{-i}) = \\
&\sum_{n=1, n\neq i}^{N} U_n^{loc} + \theta_i \ln\Big(1 + ((\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) - K_i^{vec})^+\Big) \\
&- (1-\theta_i)\rho_{vec} f_i^{vec} - \sum_{n=1}^{N} (\ln(1 + ((\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) \\
&- K_n^{loc})^+)) - \varpi I((\Lambda_n^E e_n^{loc(max)} + \Lambda_n^T t_n^{max}) < K_n^{loc})) \\
&= U(d_i^{vec}, d_{-i}) - U(d_i^{loc}, d_{-i}),
\end{aligned} \tag{23}$$

A potential function is introduced, as expressed in (22) [47], [48], to prove the existence of NE in the task offloading strategy game. Then, for each vehicle $i (i \in \mathcal{N}), d_{-i} \in \Pi_{j\neq i}\mathcal{D}_j$, the potential function is illustrated to meet (21) when $i$ updates its current offloading decision $d_i^{'}$ to $d_i$. It can be shown from the findings of (22) and (23) that the potential function $\psi(d)$ always satisfies (21) for any of two offloading decisions of vehicle $i$. Thus, the task offloading strategy game is an exact potential game.

Moreover, the potential game updates the task offloading strategies after the resource allocation stage until NE is achieved. Once all vehicles take the offloading decisions, the computation resource allocation of VEC computing needs optimization to maximize all offloading vehicles' utility. Through mutual iteration, the system enters into a steady-state and achieves the near-optimal solution.

### B. Computation Resource Allocation

Here, the computation resource allocation is determined optimally for vehicles to offload their tasks. This scheme aims to maximize the vehicles' utility by offloading the tasks to the VEC server. While the vehicle tasks are offloaded to the

VEC, the resource allocation must be optimally acquired in the following ways:

$$
\begin{aligned}
&\max_{\mathcal{F}} \sum_{i\in\mathcal{N}_o} \theta_i \ln(1+((\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) - K_i^{vec})^+) \\
&- (1-\theta_i)\rho_{vec} f_i^{vec} \\
&s.t. \quad C7: t_i^{vec} \le t_i^{ptd}, \\
&\qquad C2, C3 \quad \forall i \in \mathcal{N}_o,
\end{aligned} \tag{24}
$$

*Lemma 2:* The above problem presented in (24) is convex.
*Proof:* solved in Appendix A.
As (24) is convex, by using the partial Lagrange Function, the problem can be expressed as:

$$
\begin{aligned}
&L(\mathcal{F},\Omega) = \\
&\sum_{i\in N_m} \theta_i \ln(1+((\Lambda_i^E e_i^{vec(max)} + \Lambda_i^T t_i^{ptd}) - K_i^{vec})^+) \\
&- (1-\theta_i)\rho_{vec} f_i^{vec} + \Omega(\sum_{i=1}^{N} f_i^{vec} - F_i^{vec}),
\end{aligned} \tag{25}
$$

where $\Omega$ is the Lagrange multiplier associated with the VEC server's computing resource constraint, and $\Omega \ge 0$.

Finally, using the KKT conditions for optimally achieving computation resource allocation $\mathcal{F}^*$. Through differentiating $L(\mathcal{F},\Omega)$ referring to $f_i^{vec}(i \in \mathcal{N}_o)$, and makes it equal 0, it can be achieved as describe in (26). Next, we employ a bisection algorithm to obtain near-optimal resource allocation solution $f_i^{vec*}$, as shown in Algorithm 1.

### *C. The Distributed MACTER Algorithm*

In this section, we design a distributed MACTER Algorithm by taking advantage of the task offloading strategy game and computation resource allocation for our proposed scheme. The objective of the proposed algorithm is to maximize computation efficiency.

In Algorithm 1, We show how resources are allocated and offloading decisions are taken. The vehicles decide to carry out the computation task by selecting an offloading strategy, such as $d_i^{vec} = 1$ for VEC computing or $d_i^{loc} = 1$ for computing the task locally in the vehicle. Algorithm 1's Lines 1-10 represent the computation resource allocation part. For getting the near-optimal computation resource allocation solution, the Lagrange multiplier technique and the bisection method are leveraged. Where the KKT conditions are applied for obtaining resource allocation $\mathcal{F}^*$. By differentiating $L(\mathcal{F},\Omega)$ as represented in (25) according to $f_i^{vec}(i \in \mathcal{N}_o)$, and make it equal to 0, it can be carried out as (26). Then, we applied a bisection algorithm to achieve the resource allocation solution $f_i^{vec*}$. In addition, the second part of Algorithm 1 from Lines 11-23 represents the offloading strategy. For all vehicles, by calculating $f_i^{vec*}$ based on (26), $K_i^{loc}$ and $U_i^{loc}$ according to (13) and (15), and $K_i^{vec}$ and $U_i^{vec}$ by putting $f_i^{vec*}$ taken from the first part of the Algorithm 1 into (10) and (16), respectively. Then, conditional to the maximum tolerable delay and maximum energy consumption, the offloading decisions are made concerning the utility function. Finally, the Algorithm 1 gives the output as resource allocation $\mathcal{F}^*$ and offloading strategy $U_i^{opt*}$.

**Algorithm 1:** Resource Allocation and Offloading Strategy

**Input :** Vehicles $\mathcal{N} = \{1, 2, ..., N\}$, task $\varphi_i = \{C_i, \alpha_i^{in}, t_i^{max}\}, i \in \mathcal{N}$, Maximum tolerance $\epsilon > 0$ , $\Omega^{min} = \Omega^{max} = \Omega^{bound}$

**Output:** $\mathcal{F}^* = \{f_1^{vec*}, f_2^{vec*}, ..., f_n^{vec*}\}$ and $U_i^{opt^*}$

```
1  while Ω^max − Ω^min > ε do
2      Ω = (Ω^min + Ω^max)/2, and compute f_i^vec
3      according to put Ω into (26)
4      a = 1+(Λ_i^E e_i^vec(max) + Λ_i^T t_i^ptd) − (Λ_i^E e_i^vec + Λ_i^T α_i^in/γ_i)
       b = − C_i θ_i / ((1−θ_i)ρ_vec + Ω*)
```

$$
f_i^{vec*} = \frac{C_i + \sqrt{C_i^2 - 4ab}}{2(a)} \tag{26}
$$

```
       if Σ_{i=1}^{N} f_i^vec < F_i^vec then
5          Ω^max = Ω
6      else
7          Ω^min = Ω
8      end
9  end
10 By putting Ω into (26), we obtain the optimal
     computation resource allocation
11 for all Vehicle i ∈ N do
12     Calculate f_i^vec* by (26), calculate K_i^loc and U_i^loc
         according to (13) and (15) respectively, calculate
         K_i^vec and U_i^vec by putting f_i^vec* into (10) and
         (16) respectively.
13     if (Λ_n^E e_n^loc(max) + Λ_n^T t_n^max) < K_i^loc &&
         (Λ_i^E e_i^vec(max) + Λ_i^T t_i^ptd) < K_i^vec then
14         U_i^opt = −∞, d_i^loc = 0, d_i^vec = 0
15     else
16         if U_i^vec > U_i^loc then
17             d_i^loc = 0, d_i^vec = 1
18         else
19             d_i^loc = 1, d_i^vec = 0
20         end
21     end
22 end
23 U_i^opt* = {U_1^opt*, U_2^opt*, ..., U_N^opt*} can be derived.
```

Algorithm 2 provides a detailed description of the proposed scheme. The algorithm carries out the iterative process. An initial offloading strategy is considered at first. Subsequently, offloading strategy is updated by comparing the size of $U_i^{vec}$ and $U_i^{loc}$. The $U_i^{opt}$ is obtained from Algorithm 1. Based on the current offloading strategy, the utility of each vehicle is calculated separately in each iteration. To improve the respective utility, each vehicle then updates its offloading strategy. When no vehicle has the urge to change its offloading decision, the iterative process is terminated. Then for all vehicles, the computation efficiency is obtained. Our scheme achieves the near-optimal solutions (task offloading decision and computation resource allocation for IoV), whereby NE of task offloading decision is made based on the proposed algorithm. The NE indicates that each vehicle obtains its final offloading decision, and the vehicles will never differ from this offloading strategy. The vehicles choose the most suitable offloading strategy for task completion based on the MACTER algorithm. Finally, the Algorithm 2 provides computation efficiency as an output by achieving near-optimal computation resource allocation $\mathcal{F}^*$ and final offloading strategy $\mathcal{D}^*$.

The computational complexity of Algorithm 1 is represented as $\mathcal{O}(\log_2((\Omega^{max} - \Omega^{min}/\epsilon) + N))$. Besides, the while-loop in Algorithm 2 needs a constant number of iterations as $C$ to converge, thus the computational complexity of the MACTER scheme is $\mathcal{O}(CN(log_2((\Omega^{\max} - \Omega^{\min})/\varepsilon) + N))$.

**Algorithm 2:** The Distributed MACTER Algorithm.

**Input :** Vehicle $\mathcal{N} = \{1, 2, ...N\}$, task $\varphi_i = \{C_i, \alpha_i^{in}, t_i^{max}\}, i \in N$, vehicle speed $v \sim CN(U, \sigma^2)$, vehicle position $s_i$, initial offloading strategy $D_\Theta$

**Output:** The computation efficiency $\mathbb{E}$, the computation resource allocation $\mathcal{F}^*$, and offloading strategy $\mathcal{D}^*$

Allocate computation resource $f_i^{vec*}$ to the vehicles by Algorithm 1, calculate $U_i^{vec}$ based on (16)
**for** *each vehicle* $i \in \mathcal{N}$ **do**
  **if** $U^{loc} > U_i^{vec}$ **then**
    $d_i^{vec} = 0, d_i^{loc} = d_i^{loc}$
  **else**
    $d_i^{vec} = 1, d_i^{loc} = 0$
  **end**
  $D(t) = \{D^{loc}(t), D^{vec}(t)\}$
  Calculate $\mathbb{E}$ according to (17)
**end**
$t = t + 1$
**while** $D(t-1) \neq D_\Theta$ **do**
  $D_\Theta = D(t-1)$
  $i = 1$
  **while** $i \leq N$ **do**
    $d_i^{'} = d_i^{vec} = 1$, and compute $U_{update}(i) = U(d_i^{'}, d_{-i}(t-1))$ based on Algorithm 1
    $i = i + 1$
  **end**
  **for** *all vehicle* $i \in \mathcal{N}$ **do**
    **if** *vehicle* $i$ *has maximal* $U_{update}$, *or* $d_\Theta(i) = d_i^{vec} = 1$ **then**
      $d_i(t) = d_i^{'}$
    **else**
      $d_i(t) = d_i(t-1)$
    **end**
  **end**
  $t = t + 1$
**end**

## V. Numerical Results

This section provides numerical results and a discussion of our proposed algorithm. We assume five RSUs, each having a VEC server placed along a unidirectional road in an urban area. In our simulation, we consider the computation resource of each vehicle as $[10, 15]$ GHz. We set the input data size $[20, 60]$ KB, the service coefficient $\hbar_i$ in the range of $[0.2, 0.4]$ GHz/KB, and the weights of the task executing time $\Lambda^T$ and energy consumption $\Lambda^E$ are set as $0.5$ if not specified. For each computation task, the maximum latency constraint takes a uniform distribution $[0.2, 1]$s. The vertical distance between RSU and road is considered $e = 100$ m. We further set the cellular link and mmWave communication range to 200 m and $150$ m, respectively. We assume that RSU broadcasts beacon messages, including the computation resource information, to the vehicles in its communication range, and once the connection is established, the communication goes into unicast mode between RSU and vehicle [41]. Also, all the vehicles share their relevant information periodically with the RSU. In our considered scenario, we leveraged mmWave connectivity, since these message sizes are minimal compared to the higher bandwidth used in 5G NR-V2X, therefore, we ignore communication overhead. The simulations are performed in MATLAB by implementing the mobility model presented in Section II. The detailed setting of other simulation parameters is summarized in Table II.

TABLE II: SIMULATION PARAMETERS

| Parameter | Value |
|---|---|
| mmWave Bandwidth | 200 MHz [49] |
| Cellular link Bandwidth | 20 MHz |
| The arrival rate of vehicles | 0.1 (veh/s) |
| The coverage length of RSU | 200 m |
| The number of RSUs | 5 |
| The computation resource price for VEC | 0.03 ($/Ghz) |
| The average vehicular velocity | 40 Km/h |
| Transmission power of vehicle $p_i$ | 1.3W |
| RSU main lobe gain $\mathcal{G}_m$ | 15db |
| Vehicle main-lobe gain $\mathcal{G}_i$ | 15db |
| Path loss exponent $\zeta$ | 3.2 |
| Energy consumption per computing unit | $[2 * 10^{-10}, 2 * 10^{-6}]$w |

The proposed MACTER scheme is evaluated against the following baseline schemes.

- Computation Task Offloading and Resource Allocation (CTORA) scheme [27], that only optimizes the offloading decisions in a given computing resource.
- Computation Offloading Decision Optimization (CODO) scheme in which the tasks are executed either locally or offloaded to the VEC server. The main difference between CODO and MACTER schemes is that CODO does not consider mmWave communication.
- Heuristic Scheme [9], in which tasks are offloaded to VEC server when the energy and time constraints of a vehicle are not being satisfied while doing computation locally. This process is done without the consideration of other vehicles.

Fig. 3 represents the relationship of vehicular velocity and practical tolerance delay $t_i^{pdt}$, we consider the vehicle $i$ maximum tolerable delay as 1s. Here, we analyze the vehicular velocity from $40$ km/h to $100$ km/h. The practical tolerance delay is inversely proportional to the velocity of the vehicle. It

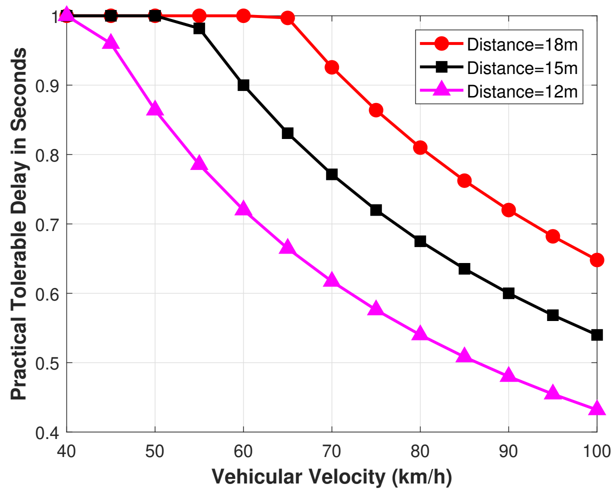



Fig. 3: The task practical tolerable delay versus various vehicular velocity in km/h.

can be observed from Fig. 3 that the time vehicle $i$ remains in the corresponding RSU becomes less as compared to the task's maximum tolerable delay, as the vehicular velocity increases. The delay of the task will be changed to $t_i^{stay}$. Conversely, when the vehicular velocity is low, the delay constraint is equal to the maximum tolerable delay for each task $t_i^{max}$. It intends to ensure smooth task transmission in the VEC network.

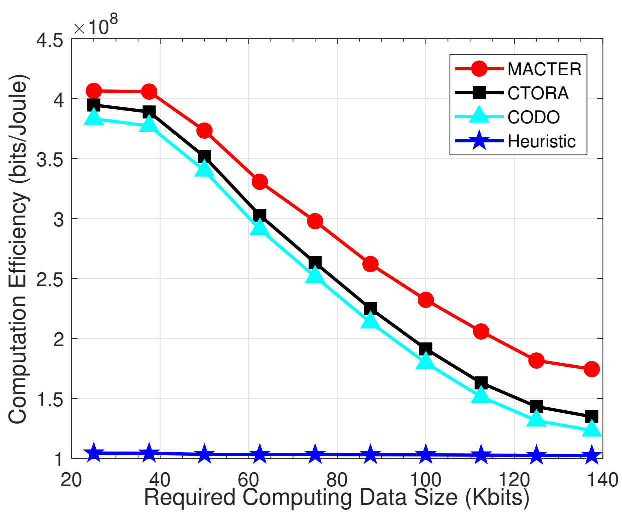



Fig. 4: The computation efficiency versus required computing data size.

Fig. 4, shows the comparison results between all four schemes, including the proposed scheme, CTORA, CODO, and Heuristic. In Fig. 4, the computation efficiency decreases with an increase in the required data bits for all the schemes. This implies that the energy required for computation is rising with the increase in the data size. The proposed algorithm evidently outperforms other schemes. Furthermore, we can note that with the small data size, the performance of the proposed scheme is closer to others except the heuristic scheme. In the heuristic approach, the offloading decision is made when the vehicle does not meet local computation time and energy constraints. The graph declines very slightly in the heuristic approach as the data grows. Moreover, the optimal choice for a relatively small data size is to compute locally. Since offloading may not be the best option as it is dependent on channel gain and available bandwidth between vehicle and RSU. Also, the energy consumption of local computing decreases more significantly than the energy consumption of offloading. Consequently, it may require a longer time and higher energy for small data. On the other hand, it can be a more suitable choice to offload it to a robust VEC server with large data size. Therefore, from the perspective of optimal resource allocation, it can be observed that it is best to process them on the VEC server for large tasks.

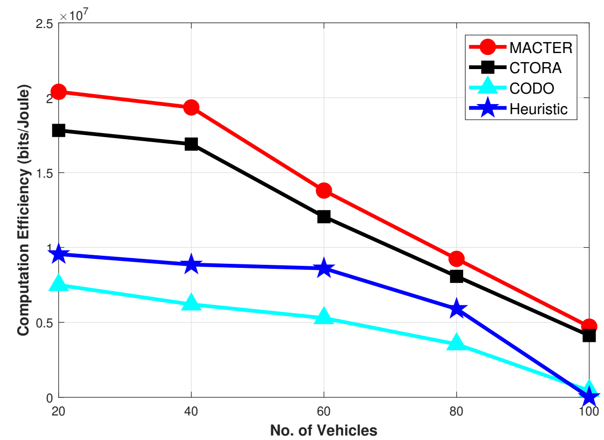
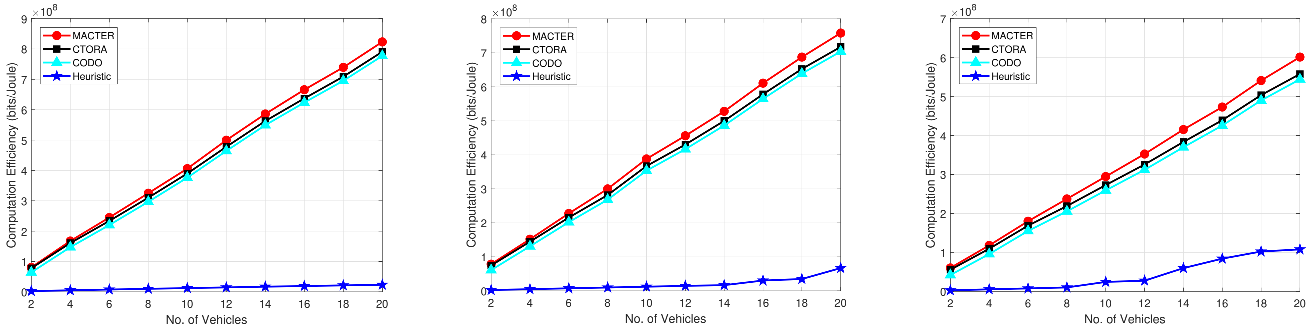



Fig. 5: The computation efficiency versus the varying number of vehicles for communication performance.

Fig. 5, illustrates the impact of communication performance on computation efficiency with the varying number of vehicles. Here, we fixed the vehicles' speed to 40 km/h and task data size to 1 kbits and considered the random location of vehicles within an RSU to examine the effect on the communication performance between vehicles and RSUs. In particular, we can see from Fig. 5 that performance degradation occurs in all the schemes as the number of vehicles gradually increases. This result is interpreted by the fact that the communication performance is affected by numerous factors, such as the task data size, vehicles' location, number of vehicles in each V2X communication technology (i.e., cellular and mmWave). Most significantly, It can also be observed that the SNR decreases with the increased vehicles' distance from the RSU as a consequence, degrades the performance. The CODO scheme performs worst because it does not consider the mmWave communication, and the heuristic scheme performance is better than CODO even when the vehicles are above 90. From the communication perspective, it can be observed that the proposed scheme gives the best performance for any number of vehicles in an RSU compared to other schemes.

Fig. 6, shows the computation efficiency versus the varying number of vehicles with different speed. To show the difference, we set the vehicle speed 30 km/h, 50 km/h, and 70 km/h, as shown in Fig. 6a, 6b, and 6c, respectively.




(a) Vehicle speed 30 km/h. (b) Vehicle speed 50 km/h. (c) Vehicle speed 70 km/h.

Fig. 6: Computation Efficiency with different speed in the case of varying the number of vehicles.

For convenience in comparison, we set the equal task size for different vehicles. While compared with the behaviors of different schemes, we observe that the proposed scheme obtains the highest computation efficiency among all the given speeds. The other schemes also increase as the no. of vehicle increase. We can observe that the computation efficiency of all schemes decreases rapidly as the vehicles' speed increases. Moreover, it can be noticed that when the speed increases, then the heuristic algorithm grows since the $t_{ptd}$ becomes lesser and vehicles stay for less time in their corresponding RSU. Therefore, vehicles prefer to compute locally.

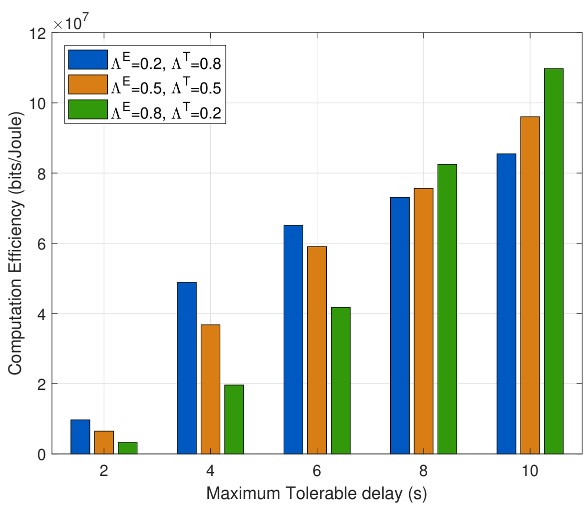



Fig. 7: The computation efficiency with the different weightage of the energy and time for maximum tolerable delay.

In Fig. 7, the computation efficiency with different energy and time weight factors are illustrated as $\Lambda^E$ and $\Lambda^T$, respectively. It can be noted that with various weight factors, the vehicles can achieve different computation efficiency. If the vehicle is running out of the battery and have sufficient maximum tolerable delay then it prefers to offload, when $\Lambda^E = 0.8$ & $\Lambda^T = 0.2$ and when $\Lambda^E = 0.2$ & $\Lambda^T = 0.8$ then time factor becomes more significant in decision making with stringent delay requirements. Moreover, when $\Lambda^E = 0.5$ & $\Lambda^T = 0.5$, then the vehicle notice both the aspects of energy and time and make the offloading decision accordingly. From Fig. 7, we can observe that the weight factor can affect the overall system performance, as well as fairness when considering the computation efficiency.

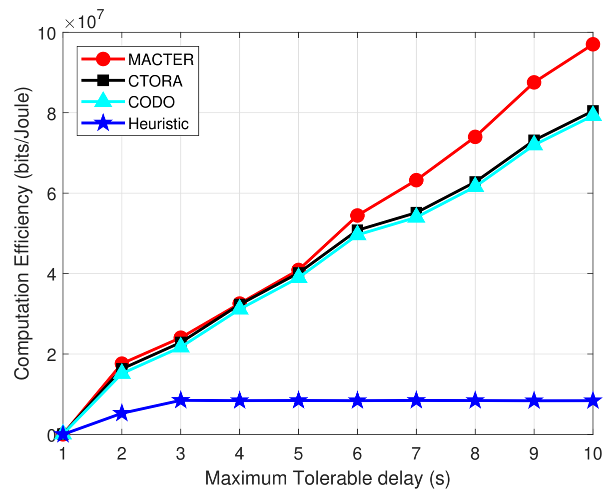



Fig. 8: The computation efficiency versus maximum tolerable delay.

In Fig. 8, we compare the computation efficiency and the maximum tolerable delay tolerance of different algorithms. When the tolerable delay rises, the total energy consumption declines, and the performance gap in all other algorithms and the MACTER algorithm grows larger. The results show that compared with the CTORA, CODO, and Heuristic algorithm, the proposed algorithm can significantly increase the computation efficiency over the considered range of $t_i^{max}$, by about 17%, 19%, 91%, respectively. Note again that the computation efficiency of the CTORA and CODO algorithms almost coincide with the estimated range of $t_i^{max}$ values. In fact, since the computation capacity of the VEC server is far greater than a vehicle and while delay tolerance is stringent, the vehicles are more willing to transfer more of their tasks to the VEC server to minimize latency. Moreover, the Heuristic approach converges when the $t_i^{max}$ equal to 3 but with much less computation efficiency compared to other schemes.

Fig. 9, presents the impacts of the VEC resource price on computation efficiency when using different algorithms. It can be observed that the entire computation efficiency of these ap-

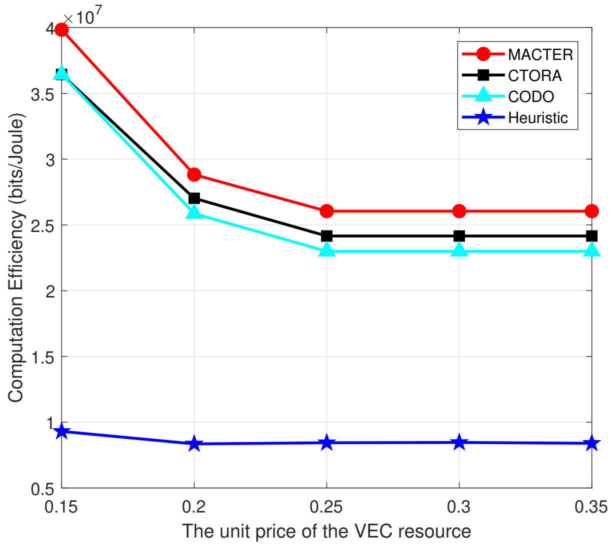



Fig. 9: The computation efficiency versus unit price of the VEC resource.

proaches drops down significantly as the price increases. Since the VEC server has more computing resources, more tasks can be offloaded to the VEC server as the density of the vehicles increases. Consequently, the computation efficiency for a large number of vehicles is more prone to be influenced by the variation in the price of the VEC resource and decreases with the increase in price. From Fig. 9, we can observe that when the price is 0.25$, the computation efficiency of all schemes converge except the heuristic scheme, where it converges at 0.2$. The reason is that according to system utility, the VEC utility decreases, as the price of VEC resources is high.

## VI. Conclusion

In this paper, we studied a computation efficiency problem, where a vehicle intends to offload its tasks to maximize the computation efficiency in terms of a tradeoff between computation time and energy consumption. We formulated the computation efficiency problem by jointly optimizing task offloading and computation resource allocation. To achieve a distributed task offloading, we utilized a game-theoretic method. We presented a mobility-aware computational efficiency based task offloading and resource allocation (MACTER) scheme and developed a distributed MACTER algorithm that provides the near-optimal solution. We also used the 5G NR-V2X communication model, i.e., cellular link and mmWave, to improve the system performance. The numerical findings revealed that the proposed algorithm can improve computation efficiency while meeting computing time and energy consumption constraints. For the future, we would extend our work with more complex scenarios in which precise information about the channel and vehicle state is unknown. To this end, we will examine how to integrate machine learning techniques by incorporating 5G NR and mmWave communications to improve long-term delay performance to further strengthen the task offloading process.


## Acknowledgment

This work was supported by the Key-Area Research and Development Program of Guangdong Province under the grant number (2020B010164002)


## Appendix A
## A Proof of Lemma 2

To demonstrate that the problem is convex in (24), we have shown that the utility function $U_i^{vec}$ is a concave function of $f_i^{vec}$. We can obtain $U_i^{vec}$ by differentiating with respect to $f_i^{vec}$,

$$\frac{\partial U_i^{vec}}{\partial f_i^{vec}} = \frac{\theta_i}{1+((\Lambda_i^E e_i^{vec(max)}+\Lambda_i^T t_i^{ptd})-K_i^{vec})}\frac{C_i}{(f_i^{vec})^2}-(1-\theta_i)\rho_{vec}, \tag{27}$$

The second-order derivative of $U_i^{vec}$ with respect to $f_i^{vec}$ is expressed as

$$\frac{\partial^2 U_i^{vec}}{\partial (f_i^{vec})^2} = \frac{\frac{\theta_i C_i}{(f_i^{vec})^3}\left(2\left(1+((\Lambda_i^E e_i^{vec(max)}+\Lambda_i^T t_i^{ptd})-K_i^{vec}\right)+\frac{C_i}{f_i^{vec}}\right)}{\left(1+((\Lambda_i^E e_i^{vec(max)}+\Lambda_i^T t_i^{ptd})-K_i^{vec})\right)^2} \tag{28}$$

where $(\Lambda_i^E e_i^{vec(max)}+\Lambda_i^T t_i^{ptd})-K_i^{vec}>0$ and $\frac{\partial^2 U_i^{vec}}{\partial (f_i^{vec})^2}<0$.

## References


[1] K. Jo, J. Kim, D. Kim, C. Jang, and M. Sunwoo, “Development of autonomous car—part ii: A case study on the implementation of an autonomous driving system based on distributed architecture,” *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5119–5132, 2015.

[2] J. Feng, Z. Liu, C. Wu, and Y. Ji, “Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling,” *IEEE vehicular technology magazine*, vol. 14, no. 1, pp. 28–36, 2018.

[3] F. Lyu, N. Cheng, H. Zhu, H. Zhou, W. Xu, M. Li, and X. S. Shen, “Intelligent context-aware communication paradigm design for iovs based on data analytics,” *IEEE Network*, vol. 32, no. 6, pp. 74–82, 2018.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[5] P. Zhang, M. Zhou, and G. Fortino, “Security and trust issues in fog computing: A survey,” *Future Generation Computer Systems*, vol. 88, pp. 16–27, 2018.

[6] 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 Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 137–12 151, 2018.

[7] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, “Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm,” *Sensors*, vol. 21, no. 3, p. 779, 2021.

[8] X. Zhang, J. Zhang, Z. Liu, Q. Cui, X. Tao, and S. Wang, “Mdp-based task offloading for vehicular edge computing under certain and uncertain transition probabilities,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3296–3309, 2020.

[9] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, “Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading,” *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, 2017.

[10] L. Silva, N. Magaia, B. Sousa, A. Kobusińska, A. Casimiro, C. X. Mavromoustakis, G. Mastorakis, and V. H. C. De Albuquerque, "Computing paradigms in emerging vehicular environments: A review," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 3, pp. 491–511, 2021.

[11] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: architecture, applications, technical issues, and future directions," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[12] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Vehicular Technology Magazine*, vol. 14, no. 3, pp. 100–106, 2019.

[13] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, and J. Li, "Collaborative learning of communication routes in edge-enabled multi-access vehicular environment," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 4, pp. 1155–1165, 2020.

[14] H. Zhou, W. Xu, Y. Bi, J. Chen, Q. Yu, and X. S. Shen, "Toward 5g spectrum sharing for immersive-experience-driven vehicular communications," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 30–37, 2017.

[15] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2018.

[16] Z. Zhou, J. Feng, Z. Chang, and X. Shen, "Energy-efficient edge computing service provisioning for vehicular networks: A consensus admm approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5087–5099, 2019.

[17] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE internet of things journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[18] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, pp. 1–11, 2020.

[19] J. Bi, H. Yuan, S. Duanmu, M. C. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet of Things Journal*, 2020.

[20] X. Chen, H. Zhang, C. Wu, S. Mao, Y. Ji, and M. Bennis, "Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4005–4018, 2018.

[21] J. Feng, Z. Liu, C. Wu, and Y. Ji, "Ave: Autonomous vehicular edge computing framework with aco-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10 660–10 675, 2017.

[22] T. Taleb, S. Dutta, A. Ksentini, M. Iqbal, and H. Flinck, "Mobile edge computing potential in making cities smarter," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 38–43, 2017.

[23] X. Li, Y. Dang, M. Aazam, X. Peng, T. Chen, and C. Chen, "Energy-efficient computation offloading in vehicular edge cloud computing," *IEEE Access*, vol. 8, pp. 37 632–37 644, 2020.

[24] Z. Zhou, H. Yu, C. Xu, Z. Chang, S. Mumtaz, and J. Rodriguez, "Begin: Big data enabled energy-efficient vehicular edge computing," *IEEE Communications Magazine*, vol. 56, no. 12, pp. 82–89, 2018.

[25] D. Ye, M. Wu, J. Kang, and R. Yu, "Optimized workload allocation in vehicular edge computing: A sequential game approach," in *International Conference on Communicatins and Networking in China*. Springer, 2017, pp. 542–551.

[26] J. Zhao, L. Wang, K.-K. Wong, M. Tao, and T. Mahmoodi, "Energy and latency control for edge computing in dense v2x networks," *arXiv preprint arXiv:1807.02311*, 2018.

[27] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[28] Y. Liu, S. Wang, J. Huang, and F. Yang, "A computation offloading algorithm based on game theory for vehicular edge networks," in *2018 IEEE International Conference on Communications (ICC)*. IEEE, 2018, pp. 1–6.

[29] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[30] N. Kumar, S. Zeadally, N. Chilamkurti, and A. Vinel, "Performance analysis of bayesian coalition game-based energy-aware virtual machine migration in vehicular mobile cloud," *IEEE Network*, vol. 29, no. 2, pp. 62–69, 2015.

[31] X. Huang, D. Ye, R. Yu, and L. Shu, "Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 426–441, 2020.

[32] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.

[33] J. Zhang, W. Xia, Z. Cheng, Q. Zou, B. Huang, F. Shen, F. Yan, and L. Shen, "An evolutionary game for joint wireless and cloud resource allocation in mobile edge computing," in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2017, pp. 1–6.

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

[35] R. Yu, X. Huang, J. Kang, J. Ding, S. Maharjan, S. Gjessing, and Y. Zhang, "Cooperative resource management in cloud-enabled vehicular networks," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 12, pp. 7938–7951, 2015.

[36] S. Yousefi, E. Altman, R. El-Azouzi, and M. Fathy, "Analytical model for connectivity in vehicular ad hoc networks," *IEEE Transactions on Vehicular Technology*, vol. 57, no. 6, pp. 3341–3356, 2008.

[37] S. Durrani, X. Zhou, and A. Chandra, "Effect of vehicle mobility on connectivity of vehicular ad hoc networks," in *2010 IEEE 72nd Vehicular Technology Conference-Fall*. IEEE, 2010, pp. 1–5.

[38] I. Rasheed and F. Hu, "Intelligent super-fast vehicle-to-everything 5g communications with predictive switching between mmwave and thz links," *Vehicular Communications*, vol. 27, p. 100303, 2021.

[39] K. Guan, G. Li, T. Kürner, A. F. Molisch, B. Peng, R. He, B. Hui, J. Kim, and Z. Zhong, "On millimeter wave and thz mobile radio channel for smart rail mobility," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 7, pp. 5658–5674, 2016.

[40] G. Naik, B. Choudhury, and J.-M. Park, "Ieee 802.11 bd & 5g nr v2x: Evolution of radio access technologies for v2x communications," *IEEE Access*, vol. 7, pp. 70 169–70 184, 2019.

[41] 3GPP, "Study on enhancement of 3gpp support for 5g v2x services (v16.2.0, release 16)," *3gpp rel 16*, no. TR 22.886, 2019.

[42] M. H. C. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Şahin, and A. Kousaridas, "A tutorial on 5g nr v2x communications," *arXiv preprint arXiv:2102.04538*, 2021.

[43] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016.

[44] H. Wang, X. Li, H. Ji, and H. Zhang, "Federated offloading scheme to minimize latency in mec-enabled vehicular networks," in *Proceeding of the Globecom Workshops*. IEEE, 2018, pp. 1–6.

[45] 3GPP, "Study on evaluation methodology of new vehicle-to-everything v2x use cases for lte and nr (release 15)," *3gpp rel 15*, no. TR 37.885, 2019.

[46] S. Raza, W. Liu, M. Ahmed, M. R. Anwar, M. A. Mirza, Q. Sun, and S. Wang, "An efficient task offloading scheme in vehicular edge computing," *Journal of Cloud Computing*, vol. 9, pp. 1–14, 2020.

[47] J. Zhang, W. Xia, F. Yan, and L. Shen, "Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing," *IEEE Access*, vol. 6, pp. 19 324–19 337, 2018.

[48] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 5, pp. 4514–4526, 2018.

[49] Z. Li, L. Xiang, X. Ge, G. Mao, and H.-C. Chao, "Latency and reliability of mmwave multi-hop v2v communications under relay selections," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9807–9821, 2020.