Hindawi
Wireless Communications and Mobile Computing
Volume 2021, Article ID 8943862, 13 pages
https://doi.org/10.1155/2021/8943862

Research Article

Joint Load Balancing and Offloading Optimization in Multiple Parked Vehicle-Assisted Edge Computing

Xinyue Hu,[1] **Xiaoke Tang**,[2] **Yantao Yu**,[1] **Sihai Qiu**,[2] **and Shiyong Chen**[1]

[1]*School of Microelectronics and Communication Engineering, Chongqing University, Chongqing 400044, China*
[2]*Beijing Smart-Chip Microelectronics Technology Co., Ltd., Beijing 100192, China*

Correspondence should be addressed to Yantao Yu; yantaoyu@cqu.edu.cn

Received 9 August 2021; Revised 26 October 2021; Accepted 30 October 2021; Published 23 November 2021

Academic Editor: Muhammad Shiraz

The introduction of mobile edge computing (MEC) in vehicular network has been a promising paradigm to improve vehicular services by offloading computation-intensive tasks to the MEC server. To avoid the overload phenomenon in MEC server, the vast idle resources of parked vehicles can be utilized to effectively relieve the computational burden on the server. Furthermore, unbalanced load allocation may cause larger latency and energy consumption. To solve the problem, the reported works preferred to allocate workload between MEC server and single parked vehicle. In this paper, a multiple parked vehicle-assisted edge computing (MPVEC) paradigm is first introduced. A joint load balancing and offloading optimization problem is formulated to minimize the system cost under delay constraint. In order to accomplish the offloading tasks, a multiple offloading node selection algorithm is proposed to select several appropriate PVs to collaborate with the MEC server in computing tasks. Furthermore, a workload allocation strategy based on dynamic game is presented to optimize the system performance with jointly considering the workload balance among computing nodes. Numerical results indicate that the offloading strategy in MPVEC scheme can significantly reduce the system cost and load balancing of the system can be achieved.

1. Introduction

As road traffic density continues to increase and traffic data explodes, the limited computing capacity of onboard terminals cannot meet the communication and computing demand of computationally intensive onboard applications [1, 2]. The introduction of mobile edge computing (MEC) has become an effective solution to the problem of resource scarcity in vehicular networks [3, 4]. Usually, MEC can enhance network resources and enable localized data processing by deploying computation and storage resources at the edge of the network close to the users [5, 6]. Compared with remote cloud computing, it can use resource-rich servers at the roadside unit (RSU) to provide users with low-latency, high-bandwidth application services [7].

However, onboard applications such as augmented reality and autonomous driving are with higher demands on data processing and storage capabilities and still require more available resources [8, 9]. When the number of offloading tasks is large, the MEC server with limited resources will be overloaded and result in less efficient task execution. Moreover, deploying massive MEC servers to augment vehicular network resources would entail huge economic and time costs, which is clearly not feasible. In order to address the problem, end devices such as vehicles and gateways with limited resources can be used as infrastructures to effectively extend the computational, communication, and storage capabilities of the edge server. For example, the idea of using vehicles as communication and computing infrastructure to collaborate with other edge devices to perform tasks has been proposed in [10], which can meet the demand for considerable communication and computation capabilities. The computing power of mobile vehicles on the road has been used to assist in offloading tasks and speed up the task execution process [11–13]. As mobile vehicles are highly dynamic, it is difficult to guarantee the stability and reliability of task offloading.

Note that the PVs with vast idle resources in the roadside and parking lots can act as static network infrastructures to enhance vehicular network [14, 15]. According to a survey, about 70% of vehicles are parked for more than 20 hours

per day [16]. And most of the vehicles in the vehicular network have been equipped with sensors, wireless devices, and onboard units, which facilitates the vehicles to establish stable and reliable wireless communication and consequently form a vehicular adhoc network (VANET) [17, 18].

In the existing studies of vehicular edge networks, the data interaction between the driving vehicles and the roadside units is mainly considered. Generally, it leads to low utilization of the PV resources. Based on the above problems, a multiple parked vehicle-assisted edge computing framework is proposed in this paper, which has multiple parked vehicles to assist the edge server to perform offloading tasks. And the total system cost is minimized under the constraint of maximum allowable delay while taking load balancing and offloading optimization into account.

The main contributions of this work are summarized as follows.

(i) The system model of multiple parked vehicle-assisted edge computing is analyzed. And the joint load balancing and offloading optimization problem is formulated to minimize the total system cost under the delay constraint. The offloading strategy is proposed to solve the optimization problem, which includes offloading node selection and workload allocation

(ii) Considering the parked probability and resource availability of PVs, a multiple offloading nodes selection algorithm is adopted to select several candidate offloading nodes among vehicles and MEC server

(iii) Considering the sequential nature of offloading decisions and the resource consumption during task execution, an efficient workload allocation strategy based on dynamic game is proposed to optimize system utility while considering load balancing

The rest of this paper is organized as follows. In Section II, related works about task offloading in vehicular networks are firstly introduced. The system model for multiple parked vehicle-assisted edge computing is described in detail in Section III. In Section IV, the workload allocation problem among multiple tasks and multiple computing nodes is modeled as a dynamic game process. In Section V, an efficient offloading strategy is proposed to solve the node selection and workload allocation problems. Simulation results for proposed scheme and the related analysis for different cases are provided in Section VI. Finally, the research work is concluded in Section VII.

2. Related Work

In the last few years, the existing work in vehicular networks is mainly utilizing MEC to provide offloading service. These research works can be divided into two main categories: one is only using MEC servers to handle task offloading requests, and the other is using remote clouds or vehicles to assist MEC servers.

2.1. Vehicular Edge Computing. Researches in the first category aim to solve the offloading problem in the vehicular work by only using edge servers. The MEC server has more computing power and can provide a large amount of resources for offloading services. The existing work mainly focused on improving the efficiency of task execution and avoiding server overload by optimizing resource allocation. For example, a collaborative computing offload and resource allocation optimization scheme, based on the scalable nature of tasks in driver assistance applications, was presented in [19]. In order to balance resource consumption and user experience with limited computing and spectrum resources, edge computing and social networking were combined to propose a new network system—vehicular social edge computing (VSEC) in [20]. Thus, the quality of service and quality of experience of drivers were improved by optimizing the available network resources. Moreover, a multipath transmission workload balancing optimization scheme was investigated in [21], which uses multipath transport to support communication between vehicles and edge nodes. In [22], the fiber-wireless (FiWi) technology was introduced to enhance vehicular network, and a SDN-based load-balancing task offloading scheme was also proposed to minimize the processing delay.

2.2. Collaborate Vehicular Edge Computing. The second category of method for handling task offloading requests is to use other infrastructure such as remote clouds, UAVs, and vehicles to collaborate with MEC servers.

2.2.1. Remote Clouds Collaborate Vehicular Edge Computing. The remote clouds are often introduced in edge computing to provide more offloading services. For instance, a two-tier offloading architecture for cloud-assisted MEC to improve system utility and computational latency by using collaborative computational offloading and resource allocation optimization schemes was discussed in [23]. A multilayer data flow processing system, i.e., EdgeFlow, was presented in [24], to integrally utilize the computing capacity throughout the whole network and optimally the transmission resource allocation to minimize the system latency. Furthermore, a cloud-based tiered vehicle edge computing offloading framework that introduces nearby backup servers to make up for the lack of computing resources of MEC servers was presented in [25]. A game theoretic algorithm was used to design the optimal multilevel offloading scheme to improve the utility of vehicles and servers.

2.2.2. Mobile Vehicles or UAVs Collaborate Vehicular Edge Computing. Moreover, many works have proposed solutions for task offloading by using mobile vehicles or UAVs to assist MEC servers. In [26], a UAV-MEC system was investigated based on the idea of utilizing the UAV as a computing node to improve the average user latency. In [27], a cooperative UAV-enabled MEC network structure was presented to collaborate UAV offloading tasks, which the long-term utility was maximized by deep reinforcement learning-based algorithms. A distributed collaborative task offloading architecture by treating mobile vehicles as edge

computing resources was discussed to guarantee low latency and application performance in [28]. A joint energy and latency cost minimization problem was formulated while using vehicles to assist task offloading. And an ECOS scheme with three phases was proposed to effectively solve the optimal problem in [29].

2.2.3. Parked Vehicles Collaborate Vehicular Edge Computing. Task offloading via UAVs and mobile vehicles is highly dynamic and lead to discontinuity in communication which is highly unstable [30]. In contrast, vehicles parked on the roadside or in parking lots are relatively static and can provide a more stable and reliable task offloading service. Thus, another recent work introduces parked vehicles to extend edge computing capabilities. For instance, serving PVs as static nodes to extend vehicular network resources and the concept of parked vehicle assistance (PVA) was proposed in [31, 32]. In addition, using PVs to assist edge servers in handling offloading tasks was presented in [33], by organizing PVs into parking clusters and abstracting them as virtual edge servers. Eventually, the task offloading performance was effectively improved by a task scheduling strategy and an associated trajectory prediction model. In [34], a three-stage contract-Stackelberg offloading incentive mechanism was developed to optimize the system utility by making full use of the large amount of free resources in the parking lot. The computing resources were also classified to provide different contracts, and the problem was solved using backward induction. In [35], the system task allocation was optimized according to the collaborative vehicle edge computing (CVEC) framework by designing a contract-based incentive mechanism to schedule PVs to handle offloading tasks. And an optimal contract that maximizes subjective utility under information asymmetry was formulated to optimize user utility.

The related works discussed above in parked vehicle-assisted vehicular network rarely consider the load balance among computing nodes or just allocate the load between MEC server and single parked vehicle. Compared with them, in this paper, a MPVEC framework with multiple parked vehicles collaborating MEC server while executing offloading tasks is presented. The computing framework with distributed characteristics increases computing capacity of task offloading and provides users with more efficient and flexible offloading options. To ensure the reliable and stable task execution, a multiple offloading node selection algorithm based on the parking behavior and resource availability is proposed to select multiple appropriate PVs to accomplish the offloading tasks. Considering that the resource states of MEC servers and PVs are time-varying during task execution, an efficient workload allocation strategy is developed to optimize system performance and keep the load balancing.

3. System Model

3.1. Network Entities. In this section, the MPVEC system with network entities is mainly composed of requesting vehicles, service provider, MEC server, and several PVs, as shown in Figure 1. More details of the function of the network entities in the system are described as follows.

Requesting vehicle: the requesting vehicle makes task offloading decisions based on the information provided by the service provider. Part of the task is processed by requesting vehicles locally, and the other part is uploaded to the nearby RSU through vehicle to infrastructure (V2I) communication and reasonably distributes the workloads to corresponding edge nodes.

Service provider: based on the computational and storage capacity of the MEC server, the service provider can collect global information, including task information as well as the computational capacity and unit energy consumption of the requesting vehicles, MEC server, and PVs. Simultaneously, according to the offloading decision, it can dispatch the MEC server and PVs to execute the corresponding workload on demand.

MEC server: the MEC server is richer in computing resources and can provide offloading services for requesting vehicles. The task requests are transmitted wired to the MEC server for processing via the RSU.

Parked vehicle: RSUs are wired to each other, and wireless connections are established between PVs and RSU via V2I communication. PVs in the parking lot can use the idle computing resources to perform offloading tasks.

3.2. System Model. As shown in Figure 1, it is assumed that a one-way road is within the coverage of RSU, and there are N requesting vehicles moving on the road. Each vehicle generates a computation task, which can be described as $D_i = \{d_i, c_i, t_i^{\max}\}$ and $i \in N = \{1, 2, \cdots, N\}$. Here, $d_i, c_i, t_i^{\max}$ denotes the data size of task, the number of CPU cycles needed for executing task, and the maximum allowable time delay of the task, respectively.

To ensure uninterrupted communication during task execution, the task offloading process needs to be completed before the vehicle leaves the RSU coverage area. Assuming that the length of the road section covered by the RSU is L, the requesting vehicle moves on the one-way road at a constant speed of v, and its position is away from the starting position of the road section by l_i. Then, the maximum allowable time delay $t_i^{\max}$ of the task can be represented as $t_i^{\max} = (L - l_i)/v$.

The task generated by the requesting vehicle can be executed locally or offloaded to edge computing nodes. The set of $J = \{1, 2, \cdots, j\}$ represents the edge computing nodes. Among them, $j = 1$ represents the MEC server, $j > 1$ represents PVs, and the offloading part of the task can be offloaded to multiple edge computing nodes for parallel processing. The parameter of $x_{ij} \in \{0, 1\}$ represents the node selection variable. If the j-th edge computing node is selected to execute the i-th offloading task, $x_{ij} = 1$ is set; otherwise, $x_{ij} = 0$. Let $k_{ij}\,(0 \le k_{ij} \le 1)$ denotes the allocation workload ratio of the i -th offloading task to the j -th edge computing node. And $(1 - \sum_{j=1}^{J} x_{ij} k_{ij})$ represents the unoffloading ratio of the i -th task and should be executed locally. As the computing resource of the system is time-varying, and the resource consumption cost of each node is different, it is a key

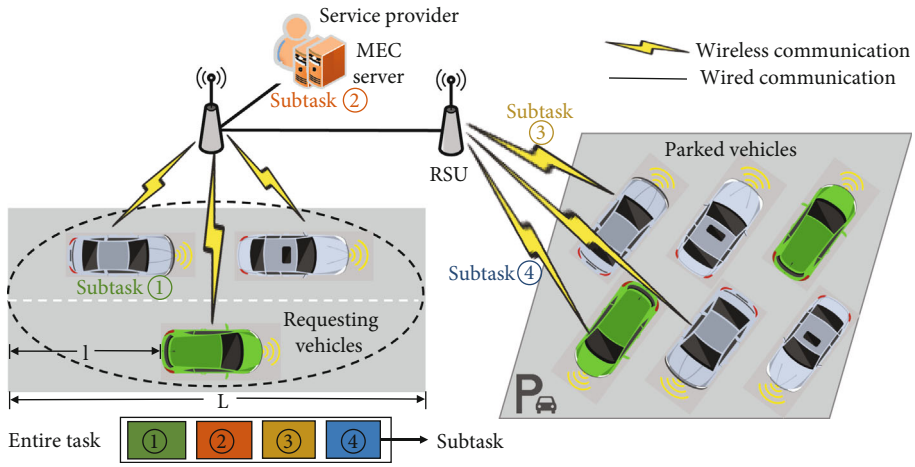


Figure 1: Multiple parked vehicle-assisted edge computing for task offloading.

challenge to balance the workload of each node while minimizing the whole system cost. The communication and computation model in the MPVEC framework will be described in the following sections. And the rest main parameters that will be used in this paper are listed in Table 1.

3.3. Communication Model. For the convenience of analysis, it is assumed that the network topology and wireless channels remain unchanged during the task execution. When the vehicle moves into the RSU coverage area, it can establish a V2I communication connection with the RSU based on IEEE 802.11p. If the task is partially or completely offloaded, the offloading part of the task is firstly transmitted to the RSU. And the MEC server, which establishes wired connection with RSU, calculates the corresponding offloading tasks. Simultaneously, the remaining offloading part is forwarded to the RSU at the parking lot, and the RSU will issue the task to the PVs for processing. Finally, the task execution result is returned. Since the size of task execution result is small, its transmission delay and energy consumption can be ignored, and only the task distribution process is considered in this paper.

(1) Requesting vehicle to RSU: when the i-th requesting vehicle who generate a computation task D_i is occurred in the coverage of RSU, the uplink transmission rate between the i-th requesting vehicle and RSU can be expressed as

$$R_i^u = B \log_2\left(1 + \frac{p_i^u h_i^u}{N_0}\right), \tag{1}$$

where $p_i{}^u$ is the transmission power of the i-th requesting vehicle, and $h_i{}^u$ is the power gain between the i-th requesting vehicle and RSU. B and N_0 are the channel bandwidth and the background noise power, respectively.

The transmission time and energy consumption of the uplink are related to the size of the task, which can be calculated by

$$\begin{aligned} T_i^u &= \sum_{j=1}^{J} \frac{x_{ij} k_{ij} d_i}{R_i^u}, \\ E_i^u &= \rho_{\text{trans}}^u \sum_{j=1}^{J} x_{ij} k_{ij} d_i, \end{aligned} \tag{2}$$

where ρ_{trans}^u is the uplink cost coefficient and represents the cost to calculate the unit data volume in the uplink.

(2) RSU to MEC server ($j = 1$): since the connection between RSU and MEC server is in a wired manner, the transmission rate is relatively high, and the data transmission time and energy consumption are negligible

(3) RSU to PV ($j > 1$): there is a RSU near the parking lot, and the RSU is connected with the roadside RSU by wire. The transmission time and energy consumption can be neglected. The RSU will send the received offloading task requests to the PVs for processing via V2I communication, and the downlink transmission rate between the RSU and the j-th PV is

$$R_{rsu,j}^d = B \log_2\left(1 + \frac{p_{rsu}^d h_{rsu,j}^d}{N_0}\right), \tag{3}$$

where p_{rsu}^d is the transmission power of the RSU, and $h_{rsu,j}^d$ is the power gain between the RSU and the j-th PV.

Furthermore, the tasks can be offloaded in parallel transmission; thus, the data transmission time of the downlink is the maximum task transmission time of each offloading part. And the transmission energy consumption is the sum of the transmission energy consumption of each offloading

Table 1: Main parameters.

Parameters	Description
d_i, c_i, $t_i^{\max}$	Task data size, task required computing resource, and the maximum allowable time delay of the task.
x_{ij}, k_{ij}	The node selection variable and the workload ratio of the i-th offloading task to the j-th edge computing node.
B, N_0	Wireless channel bandwidth and white Gaussian noise power.
p_i^u, p_i^d	Uplink and downlink transmission power.
h_i^u, $h_{rsu,j}^d$	The power gain between the i-th requesting vehicle and RSU and the power gain between RSU and the j-th PV.
f_i^{loc}, $f_{ij}^{\max}$	CPU computing power of the i-th requesting vehicle and the max CPU computing power of the j-th edge computing node.
ρ_{trans}^u, ρ_{trans}^d	Uplink and downlink cost coefficient.
$\rho_{\text{cal}}^{\text{loc}}$, $\rho_{\text{cal}}^{\text{off}}$	Energy consumption per CPU cycle of requesting vehicle and edge computing nodes.
$r_{\max}$	The maximum allowable computing resources occupancy rate of the edge computing nodes in a certain period of time.
k_i^*	The optimal strategy of the i-th requesting vehicle.
T	Time span of the time period.
$\delta(t)$, $\chi(t)$	Probability density function and accumulative distribution function of the PV parking durations t.
P_{ij}, aq_{ij}	The probability value of the j-th PV remaining parked at the execution time period of the i-th task and accumulative parking durations of the j-th PV.

part, which are defined as

$$T_i^d = \max\left\{\frac{k_{ij}d_i}{R_{rsu,j}^d}\right\},$$
$$E_i^d = \rho_{\text{trans}}^d \sum_{j=2}^{J} x_{ij}k_{ij}d_i. \tag{4}$$

We let ρ_{trans}^d as the downlink cost coefficient and represents the cost to calculate the unit data volume in the uplink.

As a result, the total transmission time and energy consumption for data transmission to the edge computing node for the offloading part of D_i are expressed, respectively, as

$$T_i^{\text{trans}} = T_i^u + T_i^d = \sum_{j=1}^{J}\frac{x_{ij}k_{ij}d_i}{R_i^u} + \max\left\{\frac{k_{ij}d_i}{R_{rsu,j}^d}\right\},$$
$$E_i^{\text{trans}} = E_i^u + E_i^d = \rho_{\text{trans}}^u \sum_{j=1}^{J} x_{ij}k_{ij}d_i + \rho_{\text{trans}}^d \sum_{j=2}^{J} x_{ij}k_{ij}d_i. \tag{5}$$

3.4. Computation Model

(1) Compute task locally: the unoffloading part of the task is calculated by the requesting vehicles locally. And the delay and energy consumption are related to the number of CPU cycles required by the task D_i, which can be calculated by

$$T_i^{\text{loc}} = \frac{\left(1 - \sum_{j=1}^{J} x_{ij}k_{ij}\right)c_i}{f_i^{loc}},$$
$$E_i^{\text{loc}} = \rho_{\text{cal}}^{\text{loc}}\left(1 - \sum_{j=1}^{J} x_{ij}k_{ij}\right)c_i, \tag{6}$$

where f_i^{loc} is the computing capability of the i-th requesting vehicle, and $\rho_{\text{cal}}^{\text{loc}}$ is the energy consumption required to calculate the unit CPU cycle.

(2) Compute task by edge computing nodes: when the task D_i is offloaded partially to the $j-$th ($j \in J$) edge computing node, the task processing delay is related to the computing capability of the edge computing node

The computing resources of the j -th edge node are limited and changed with time during the task D_i execution, which can be described as $f_{ij} \in [0, f_{ij}^{\max}]$, and $f_{ij}^{\max}$ is the max computing power of the j-th edge node.

When the computing resources occupancy rate of the edge nodes in a certain period of time is greater than its own threshold of $r_{\max}$, the tasks in the node will not be processed in parallel and need to be stored in the waiting queue and executed in sequence according to the delay constraint. Therefore, the task processing delay at the edge node mainly includes two parts: task calculation time and task waiting time, which can be written as

$$T_{ij} = \frac{k_{ij}c_i}{f_{ij}} + T_{ij}^{\text{wait}},$$
$$E_{ij} = \rho_{\text{cal}}^{\text{off}}k_{ij}c_{ij}. \tag{7}$$

Let $\rho_{\text{cal}}^{\text{off}}$ represents the energy consumption required to calculate the unit CPU cycle of the edge node, where the $\rho_{\text{cal}}^{\text{off}}$ of the PV is smaller than that of the MEC server.

The offloading part of the task request D_i generated by the i -th requesting vehicle can be processed in parallel by multiple edge computing nodes. Therefore, the task offloading delay is mainly composed of the task transmission delay and the task processing delay. And the largest processing latency among edge computing nodes is used as the task offloading latency

$$T_i^{off} = T_i^{\text{trans}} + \max\left\{T_{ij}\right\}. \tag{8}$$

Task offloading energy consumption is the sum of the transmission energy consumption and processing energy consumption of each offloading part

$$E_i^{off} = E_i^{trans} + \sum_{j=1}^{J} E_{ij} \tag{9}$$

3.5. Problem Formulation. For the task D_i generated by the i -th requesting vehicle, there is a delay and energy consumption during processing, which mainly contain two aspects: the local processing part and the offloading processing part. Due to the parallel processing of tasks, the total task processing latency is the maximum latency for local processing and task offloading processing, which can be expressed as

$$T_i = \max\left\{T_i^{\text{loc}}, T_i^{off}\right\}. \tag{10}$$

The total energy consumption of the task processing can be written as

$$E_i = E_i^{\text{loc}} + E_i^{\text{off}}. \tag{11}$$

A cost function for task D_i is defined as the combination of executing time and energy consumption.

$$U_i = \alpha T_i + \beta E_i, \tag{12}$$

where $\alpha + \beta = 1$. The goal in this paper is to minimize the whole cost of all distributed tasks with the latency constraint, while joint consider load balancing and offloading decision. The optimizing problem for all tasks can be formulated as

$$\min_{\{x_{ij},k_{ij}\}} U = \min_{\{x_{ij},k_{ij}\}} \sum_{i=1}^{N} U_i, \tag{13}$$

$$\text{s.t} \quad 0 \le k_{ij} \le 1, \forall i \in N, j \in J, \tag{14}$$

$$T_i \le t_i^{\max}, \forall i \in N, \tag{15}$$

$$x_{ij} \in \{0, 1\}, \forall i \in N, j \in J, \tag{16}$$

$$0 \le \sum_{j=1}^{J} x_{ij} k_{ij} \le 1, \forall i \in N, j \in J, \tag{17}$$

$$\alpha + \beta = 1, \alpha \ge 0, \beta \ge 0. \tag{18}$$

In the above optimization model, constraint (14) denotes the workload ratio k_{ij} is a continuous variable, which cannot exceed to 1. And each requesting vehicle can offload its task to multiple computing nodes according to the workload ratio of k_{ij}. Constraint (15) ensures that the task execution delay cannot exceed the maximum allowable task delay $t_i^{\max}$. Constraint (16) indicates the offloading node selection variable, and if the i-th requesting vehicle selects the j-th edge computing node to offload part of the task, then $x_{ij} = 1$; otherwise, $x_{ij} = 0$. Constraint (17) presents the total offloading task of the i-th requesting vehicle cannot exceed to 1 and makes the problem a mixed integer nonlinear optimization problem. In constraint (18), α and β are the weights of time delay and energy consumption in the total cost, respectively, which can be dynamically adjusted according to the task type to meet the computing requirements of different tasks.

4. Multitask Multinode Dynamic Game

In this section, the workload allocation for multiple tasks in multiple computing nodes is modeled as a dynamic game process. Considering that offloading decisions are sequential, the requesting vehicle who makes the former decision will have an impact on the requesting vehicle who makes the subsequent decision. Thus, to ensure the sequential rationality of the game process, each requesting vehicle is required to make the optimal decision, so that the overall strategy of the system is optimal.

The service provider in the proposed framework can provide requesting vehicles with global information (including the available computing capability of edge nodes and task queuing sequence). To optimize the total system cost of task execution, sequential decisions for different requesting vehicles are made to offload part or all tasks to edge computing nodes. At the same time, both sides of the game complete distributed autonomous decision making in the game process, which can obtain the optimal utility and effectively relieve the computational pressure of the MEC server.

The dynamic game process with multiple tasks and multiple computing nodes can be defined by $G(N, K, U)$, while the three elements of the game can be described as

(1) $N = \{1, 2, \cdots, i, \cdots, n\}$ represents the requesting vehicle players in the game that generates the tasks and makes task offloading decisions

(2) $K_n = \{k_1, k_2, \cdots, k_i, \cdots, k_n\}$ means the task offloading decision of the requesting vehicle players, where $k_i = \{k_{i1}, k_{i2}, \cdots, k_{ij}\}$. And k_{ij} represents the workload proportion of task D_i performed by j -th the edge node

(3) The cost function U_i represents the cost required for requesting vehicle players to perform tasks, including task execution time and energy consumption

Therefore, the offloading decision for requesting vehicle players other than the i-th requesting vehicle player can be described as $k_{-i} = \{k_1, \cdots, k_{i-1}, k_{i+1}, \cdots, k_n\}$, and the i-th player needs to choose a strategy to minimize the task execution time and energy consumption, which can be expressed as

$$\min_{k_i \in (0,1)} U_i(k_i, k_{-i}), \forall i \in N. \tag{19}$$

Next, the existence of the Nash equilibrium point in the dynamic game process is discussed.

Definition 1. There exists a strategy set $K_n^* = \{k_1^*, k_2^*, \cdots, k_i^*, \cdots, k_n^*\}$ in the dynamic game $G(N, K, U)$ and if

$$U_i(k_i^*, k_{-i}^*) \le U_i(k_i, k_{-i}^*), \forall k_i \in K_n, \tag{20}$$

then strategy set $K_n{}^*$ is the Nash equilibrium of game G. At the Nash equilibrium point, it is impossible for any player to change the strategy to obtain greater utility; that is, each requesting vehicle has made the optimal offloading decision to minimize the task execution cost.

Meanwhile, for the i -th requesting vehicle, in order to minimize its own cost, the optimal strategy k_i^* needs to be obtained by solving the following problem.

$$\mu(k_i) = \arg \min_{\{k_i\}} U_i = \alpha T_i + \beta E_i. \tag{21}$$

The optimal strategy of k_i^* can be obtained by solving the following formula:

$$\frac{\partial^2 U_i(k_i)}{\partial^2 k_i} = 0. \tag{22}$$

It can be easily concluded that the formula for solving the optimal strategy is convex, and there is an optimal solution. Hence, there is an optimal strategy in the dynamic game process between multiple tasks and multiple computing nodes.

5. Efficient Workload Allocation Strategy

In this section, an efficient offloading strategy is proposed to minimize the total cost of task execution, which joint consider load balancing and offloading optimization. In the task scheduling problem formulated in this paper, the value of the offloading selection variable is 0 or 1, while the task offloading ratio can be any value between 0 and 1. Therefore, the optimization problem is a mixed integer nonlinear optimization problem. We divide it into two subproblems to solve, namely, offloading node selection and workload allocation.

5.1. Offloading Node Selection. Different form the reported works that mainly use single computing node to collaborate MEC server in task processing, in this paper, the task D_i generated by the i -th requesting vehicle is considered to decompose into multiple subtasks, and then it is offloaded to multiple edge computing nodes for joint execution. And a multiple offloading nodes selection algorithm is designed to select several appropriate computing nodes for parallel processing tasks, which is described in Algorithm 1.

It is assumed that the maximum computing power of the i-th requesting vehicle and the j-th edge computing node are $f_{ij}^{\max}$ and f_i^{loc}, respectively. The maximum computing power of each node is fixed, but the computing resources of each node change dynamically during task execution process. Part of the computing resources are occupied during task execution, and released after the task execution is completed. When the i -th requesting vehicle selects offloading nodes, the MEC server must be used as one of the offloading nodes to prevent overloading at the PVs, considering that it can provide strong computing power. As a result, the appropriate offloading nodes are mainly selected among the PVs within the coverage area of RSU.

The appropriate offloading nodes in the PVs are selected to minimize the system cost by evaluating the execution cost of subtasks. As shown in Algorithm 1, the computation task is first divided into several subtasks with equal size, and the workload ratio of the single offloading task ω is set to 0.1. Then, the local and each edge node processing cost increment required for this subtask Δu_i^{loc} and Δu_{ij} is calculated and compared according to equation (12). The additional waiting time $\mathrm{T}_{\mathrm{ij}}^{\mathrm{wait}}$ caused by resource consumption is also considered. The PV that satisfies the parking probability constraint $P_{ij} \ge P_{th}$ will be selected while its cost increment is lower than the local $(\Delta u_{ij} < \Delta u_{loc})$. Simultaneously, if the j-th edge node is selected to executing part of the task D_i,$x_i{}_j$ is set to 1, and the workload ratio k_{ij} and computing power f_{ij} will be updated during task execution.

During the task scheduling process, the i-th requesting vehicle evaluates the current resource availability status of the j-th PV, which can be described by the probability value P_{ij} of the j-th PV remaining parked state at the execution time period of the i-th task [16]. And the P_{ij} can be calculated by

$$P_{ij} = \int_{aq_{ij}+T}^{t_{\max}} \frac{\delta(t)}{1 - \chi\left(aq_{ij}\right)} dt = \frac{1 - \chi\left(aq_{ij} + T\right)}{1 - \chi\left(aq_{ij}\right)}, \tag{23}$$

where $t \in [0, t_{\max}]$ indicates the parking durations, $\delta(t)$ denotes the probability density function of the j-th PV parking duration, and $\chi(t)$ is the cumulative distribution function of $\delta(t)$. T is time span of the time period. The q_{ij} denotes the time interval detecting the parking behavior of the j-th PV, and the parameter of a is constant. The accumulative parking durations until now is recorded as aq_{ij}. Thus, the probability that the PV will continue to stay parked for at least T time slots can be predicted.

When the PV stays for a specified period of time with a higher probability, it can provide more stable and reliable resources for task execution. If the task is assigned to the

Input: Task $D_i = \{d_i, c_i, t_i^{\max}\}$; the offloading task workload ω; the computing power f_i^{lo}, f_{ij}; the parking probability P_{ij}.
Output: The node selection variable x_{ij} (j =1 denotes MEC server, j >1 denotes PV).
1: **Initialization:** $\Delta u = 0$, $\Delta t = 0$, $\Delta e = 0$ and $x_{i1} = 1$.
2: **calculate** the execution time T_i^{loc} and T_{ij}, the energy consumption E_i^{loc} and E_{ij}.
3: **set** $\Delta u_i^{loc} = T_i^{loc} + E_i^{loc}$
4: **if** $k_{ij} \geq r_{\max} f_{ij}^{\max}$ **then**
5: **Calculate** T_{ij}^{wait}
6: **set** $\Delta t_{ij} = T_{ij} + T_{ij}^{wait}$
7: **else**
8: **set** $\Delta t_{ij} = T_{ij}$
9: **end if**
10: **set** $\Delta e_{ij} = E_{ij}$
11: **then** calculate $\Delta u_{ij} = \Delta t_{ij} + \Delta e_{ij}$
12: **if** $P_{ij} \geq P_{th}$ and $\Delta u_{ij} < \Delta u_{loc}$ **then**
13: **set** x_{ij} =1
14: **update** $k_{ij} = k_{ij} + \omega$, $f_{ij} = f_{ij}$ - ω
15: **else**
16: **set** $x_{ij} = 0$
17: **end if**
18: **return** x_{ij}

Algorithm 1: Multiple offloading node selection algorithm.

PV, the extra task retransmission overhead caused by the departure of the PV can be effectively avoided. Thus, the probability that the PV keeps parked state during the task execution period is used as an important indicator to measure the availability of PV resources.

In Algorithm 1, the PVs satisfying $P_{ij} \geq P_{th}$ can be hopefully selected as offloading nodes to execute the workload, where P_{th} is a predefined threshold value in the PV selection process. Furthermore, when selecting a suitable PV as an offloading node, it is necessary to consider the computing power of the corresponding PV itself and the energy consumption per unit. The PVs that satisfy both the parking probability constraint of $P_{ij} \geq P_{th}$, and the less task executing cost of $\Delta u_{ij} < \Delta u_{loc}$ will be selected as the candidate offloading nodes. According to Algorithm 1, stable and reliable offloading nodes can be obtained to meet the task workload allocation requirements.

5.2. Workload Allocation. The workload allocation between multiple tasks and multiple nodes is modeled as a dynamic game process. Since the requesting vehicles can obtain global information according to the service provider, the game process can be regarded as a complete information game. The requesting vehicles need to make sequential decisions based on the priorities defined by task delay constraints. Considering the resource consumption in the system, a workload allocation algorithm based on dynamic game is proposed, and the backward induction is used to assist requesting vehicles in formulating their strategies. When all requesting vehicles make the optimal decision, the Nash Equilibrium is reached and the game ends. The specific steps are described in Algorithm 2 as follows.

Assuming that the task set generated by the requesting vehicles has been prioritized according to the delay constraint, that can be given as $t_1^{\max} < t_2^{\max} < \cdots < t_i^{\max}, i \in N$. According to the proposed algorithm, the requesting vehicles allocate the workload among the local and the selected offloading nodes and makes offloading decisions in turn.

Among them, the $(i+1)$-th requesting vehicle who makes the later decision develops an offloading strategy k_{i+1} based on the former decision k_i of the i -th requesting vehicle, and then it feeds the developed strategy k_{i+1} to the i -th requesting vehicle. If the former i -th requesting vehicle has a lower-cost strategy k'_i in this case, the strategy k_i is updated and the later strategy k_{i+1} changes accordingly. Iteration keeps until the strategies and costs are both no longer changed, and then the optimal solution k_i^* and k_{i+1}^* are obtained. This step is repeated until all requesting vehicles obtain the optimal strategy $K_n^* = \{k_1^*, k_2^*, \cdots, k_i^*, \cdots, k_n^*\}$, which results in the minimum total system cost and joint consider the load balancing.

6. Numerical Results

In this section, the performance of the proposed MPVEC scheme through numerical research is evaluated. By formulating an efficient offloading strategy, the requesting vehicles allocate the task requests to the local and multiple edge computing nodes for joint execution.

6.1. Parameter Setting. We consider a unidirectional road with a section of length $L = 600$ m in the coverage of RSU, and the RSU is equipped with a MEC server which can provide offloading services. On the road section, there are [10-50] requesting vehicles driving at a constant speed of $v =$ 40 km/h, and each vehicle generates a delay-sensitive task request. And there are [5–15] vehicles parking in the parking lot nearby, which can provide idle resources. The data size of

Input: The task $D_i = \{d_i, c_i, t_i^{\max}\}$, $i \in N$; the node selection variable x_{ij}; the offloading task workload ω.
Output: The final workload strategy $K_n^* = \{k_1^*, k_2^*, \cdots, k_i^*, \cdots, k_n^*\}$, $k_i^* = \{k_{i1}, k_{i2}, \cdots, k_{ij}\}$, $k_{ij} \in [0, 1]$, let $j = 0$ denotes local, $j > 0$ denotes edge node.
Initialization: $k_{ij} = 0$, $u_i = 0$.
Step 1. Workload allocation between nodes.
for each task D_i
  while $\omega_i < c_i$ **do**
    calculate the incremental cost of processing unit-sized tasks locally $\Delta u_{i0} = T_i^{loc} + E_i^{loc}$
    if $x_{ij} \neq 0$ **then**
      calculate $\Delta u_{ij} = T_{ij} + T_{ij}^{wait} + E_{ij}$
    end if
    compare $\Delta u_{i0}, \Delta u_{i1}, \cdots, \Delta u_{ij}$
    allocate subtask to node j with the smallest cost increment **then** $k_{ij} = k_{ij} + \omega$. $u_i = u_i + \Delta u_{ij}$
    set $\omega_i = \omega_i + \omega$
  end
  set $k_i = \{k_{i0}, k_{i1}, k_{i2}, \ldots, k_{ij}\}$
Step 2. Iterative to NE based on backward induction.
  let $i = i + 1$ and repeat step 1
  calculate $k_{i+1} = \{k_{i+1,0}, k_{i+1,1}, k_{i+1,2}, \ldots, k_{i+1,j}\}$, u_{i+1}
  send k_{i+1} to user i and repeat step 1
  calculate $k_i^{'}$, $u_i^{'}$
  If $u_i^{'} < u_i$ **then**
    update $k_i = k_i^{'}$
    Send k_i to user $i + 1$ and repeat step 1
    n = n + 1
    calculate $k_{i+1}^{'}$, $u_{i+1}^{'}$
    repeat step 2 until $u_{i+1}^{(n)} = u_{i+1}^{(n-1)}$ and $u_i^{(n)} = u_i^{(n-1)}$
    set $k_i^* = k_i^{(n)}$ and $k_{i+1}^* = k_{i+1}^{(n)}$
  end if
  return k_i^*, k_{i+1}^*, u_i, u_{i+1}

Algorithm 2: Workload allocation algorithm based on dynamic game.

the task is $d_i = [100, 1000]$ KB, the number of CPU cycles required for computing c_i is [500,1500] megacycles, and the maximum allowable task delay $t_i^{\max} = (L - l_i)/v$ is related to the location of the moving vehicle. We set the location $l_i = [0, 300]$ m, where the requesting vehicle is located to ensure that the vehicle can complete the task offloading process before leaving the RSU coverage area.

In addition, the probability density function of parking durations of these PVs $\delta(t)$ is formulated by [16], and the parking probability P_{ij} can be calculated according to equation (23). In the simulation, the requesting vehicles prefer to choose the PV as the candidate offloading node if its parking probability is larger than 0.85. More simulation parameters are shown in Table 2.

6.2. Performance Comparison. In this section, the proposed offloading strategy in MPVEC is simulated, and the effectiveness and feasibility of the proposed scheme is evaluated under the same or different parameters and compared with the following schemes.

(1) All task requests generated by the requesting vehicles are computed locally (local computing, LC)

(2) There is only one MEC server in the system to provide offloading services, and there is no PV to assist in the computation. The workload is allocated between local and MEC server (no parked vehicles, NP)

(3) The system has one MEC server and multiple PVs to provide offloading services, but tasks can only be offloaded to single node for processing (single node computing, SNC)

In Figure 2, the system cost of different offloading scenarios with the same parameters is compared. We set the number of tasks generated by the requesting vehicle is $N = 10$, the data size of the requesting task is equal to 500 KB, and the number of required CPU cycles is equal to 1000 megacycles. And there is one MEC server and 10 PVs in the scenario to provide offloading services. As is shown in Figure 2, the requesting vehicle generates the largest system cost when choosing LC scheme, due to that the requesting vehicle itself has weak computing power and generates large computing latency. Compared with the LC scheme, the tasks can be offloaded to the MEC server and PVs, which have much larger resources than the requesting vehicles. Hence, the system cost of the other three schemes cut down as a result.

Table 2: Simulation parameters.

Parameters	Description	Value
d_i	Task data size (KB)	[100, 1000]
c_i	Task required computing resource (megacycles)	[500, 1500]
B	Wireless channel bandwidth (MHz)	10
p_i^u	Uplink transmission power (W)	1
p_i^d	Downlink transmission power (W)	5
h_i^u, $h_{rsu,j}^d$	Power gains	1
N_0	White Gaussian noise power (dBm)	-100
f_i^{loc}	Max computing power of requesting vehicle (GHz)	[0.5, 1]
$f_{ij}^{\max}$	Max computing power of edge node (GHz)	8, [2, 3]
ρ_{trans}^u	Uplink cost coefficient (J/KB)	1×10^{-4}
ρ_{trans}^d	Downlink cost coefficient (J/KB)	1×10^{-4}
$\rho_{\text{cal}}^{\text{loc}}$	Energy consumption per CPU cycle of requesting vehicle (J/megacycles)	1.2×10^{-3}
$\rho_{\text{cal}}^{\text{off}}$	Energy consumption per CPU cycle of edge node (J/megacycles)	2×10^{-3} $[1, 2] \times 10^{-3}$
α, β	The weights of time delay and energy consumption in the total cost	0.5, 0.5
P_{th}	Predefined threshold value.	0.85

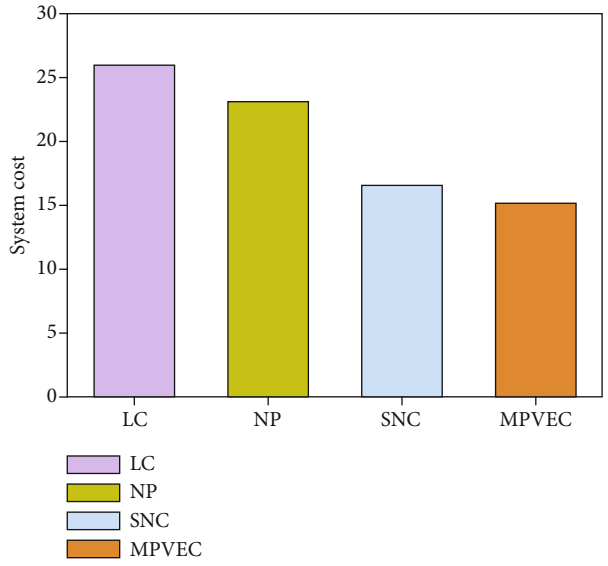


Figure 2: System cost under different scenarios.

Simultaneously, the MPVEC scheme shows the excellent performance under the same parameters, which can significantly reduce the system cost. As no PVs can provide offloading services in NP, by comparing NP with MPVEC, it clearly proves that the introduction of PVs is beneficial to the system performance. Moreover, compared with the SNC scheme, the MPVEC decomposes the tasks to multiple nodes for joint execution, the multinode distributed processing can provide more node selectivity for users, and the system cost is efficiently reduced. Numerical results show that the system cost of the proposed MPVEC is 41.5%, 34.6%, and 7.7% lower than of LC, NP, and SNC, respectively.

In Figure 3, the impact of the different number of tasks generated by the requesting vehicle on the system cost is illustrated. With the increasing number of tasks generated by the requesting vehicles, the system cost of all schemes presents an upward trend. Due to the limited resources of the computing nodes, the number of tasks in the waiting queue increases after the resource consumption reaches the threshold value of the edge nodes themselves. It will generate additional execution time costs and lead to the increasing system cost. In addition, it can be concluded from Figure 3 that the MPVEC shows better performance than other schemes under the same conditions. It is because that PVs can provide more computing resources with lower cost for offloading tasks execution. And the offloading strategy in MPVEC can effectively optimize the task execution efficiency by reasonably allocating load among computing nodes. Results indicate that the system cost of LC, NP, and SNC is 17.1%, 5.1%, and 2.6%, respectively higher than MPVEC when the number of tasks reached to 50.

In Figure 4, the impact of the number of CPU cycles required for the task of requesting vehicle generation on the system cost is illustrated. With the increase in number of CPU cycles, the computational latency and energy consumption of each node increases accordingly. Hence, the system cost has maintained an upward trend of all schemes. Moreover, it can be visualized from Figure 4 that the optimization performance of MPVEC scheme is more obvious than other schemes. When the task requires a larger amount of computing power, the task computing requirements can still be met at a lower cost in MPVEC. The numerical results indicate that the system cost increase with increasing CPU cycles is 60.1%, 32.6%, and 6.1% higher for LC, NP, and SNC than MPVEC, respectively. As a result, it can be

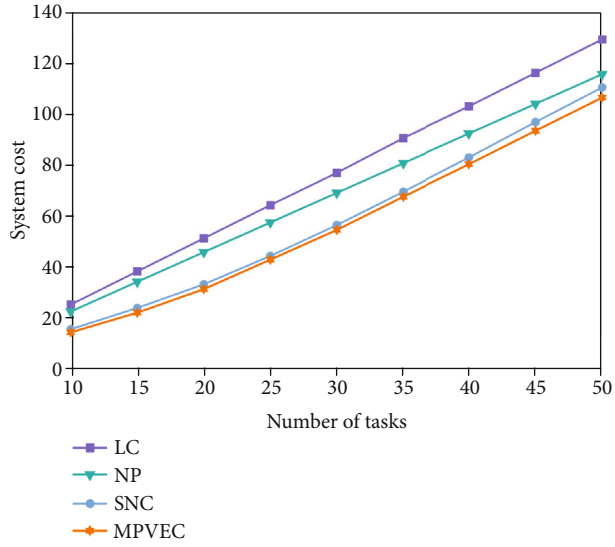


Figure 3: System cost under the change of task numbers.

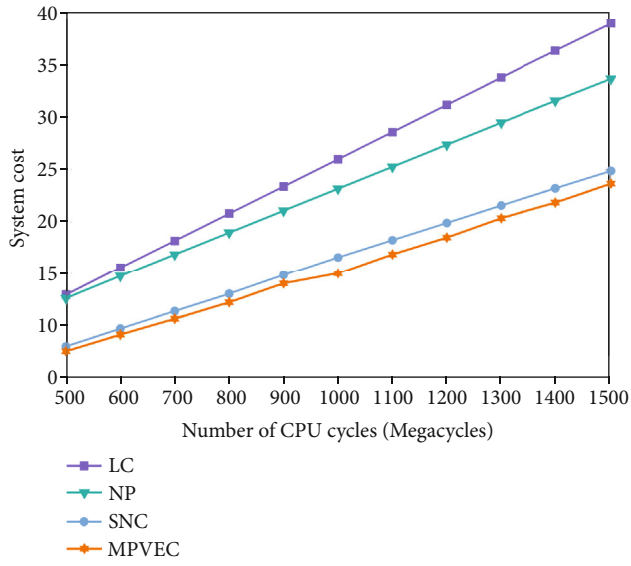


Figure 4: System cost under the change of CPU cycle numbers.

effectively proved that utilizing the large amount of idle resources of multiple PVs and allocating workload reasonably can greatly expand the edge computing capacity while providing offloading services to users at low cost.

In Figure 5, the impact of the different number of PVs on the system cost and the workload borne by the MEC server is illustrated. It can be seen the workload ratio of the MEC server and system cost decrease significantly at the beginning increase of the PV numbers. It is because that more available resources are provided for the system to accomplish offloading tasks, and the workload ratio of MEC server is reduced. It proves that the MPVEC scheme is feasible to use PVs to assist edge computing and the computational pressure on the server is considerably relieved. And the reasons for the decrease of system cost can be explained through two aspects: on the one hand, the PVs can execute

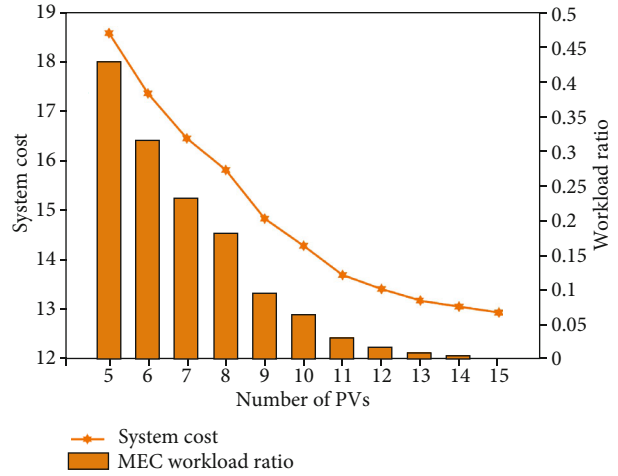
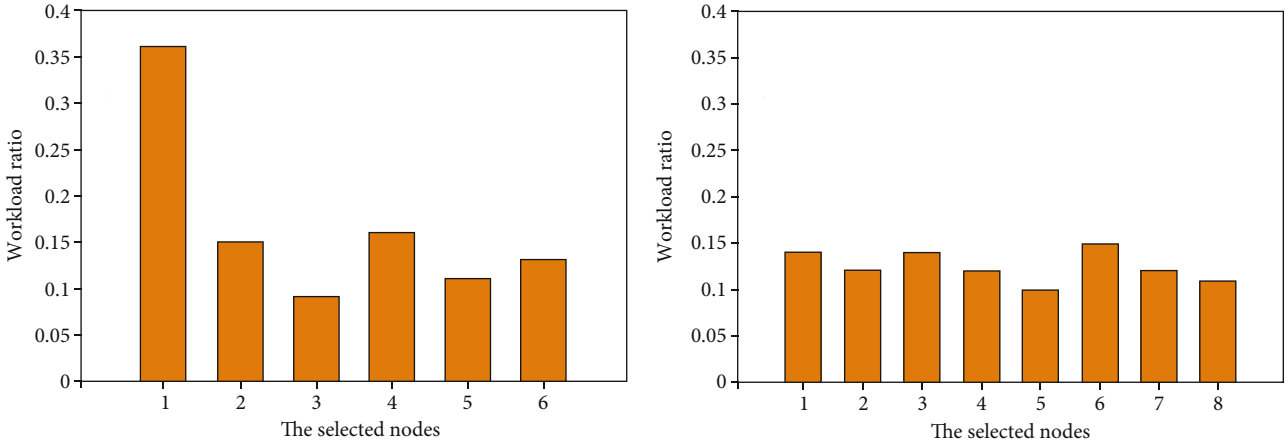


Figure 5: System cost and MEC workload ratio under the change of PV numbers.

part of the offloading tasks at a lower energy consumption per unit than the MEC server, and the system computing energy consumption is reduced as the number of the PVs becomes larger. On the other hand, as the number of PVs assisted in offloading increased, the tasks can be offloaded to more computing nodes for parallel processing, and the tasks waiting time in MEC server and the tasks computing time can be both reduced, which results in the reduction of system cost. Therefore, the multiple PV-assisted MEC can improve the system performance, and the workload of MEC server can be effectively relieved.

In Figure 6, the impact of different numbers of PVs on the load balancing of the system is illustrated. We set the number of PVs in the parking lot to 5 and 10, respectively. Comparing the load ratio of the selected computing nodes, it can be clearly seen that when the number of PVs is 5, except for individual nodes taking more workload, the workload ratio of the remaining computing nodes has a small difference, and load balancing of some nodes (except node 1) can be achieved. And when the number of PVs increases to 10, all the selected computing nodes can achieve load balancing. It can be easily concluded that through the proposed efficient offloading strategy in MPVEC, the workload of each node is reasonably distributed, which can effectively reduce the overload phenomenon and realize the system load balancing.

6.3. Complexity Analysis. The computational complexity of the proposed offloading strategy in MPVEC is $O(NM)$, where N and M are the number of requesting vehicles and the total number of computing nodes (include requesting vehicle itself and MEC server and PVs), respectively. In [28], the collaborative task offloading strategy based on a computation task and resource sharing mechanism between vehicles and edge infrastructures was reported. Its computational complexity is $O(NM)$, where N and M are the total number of edge infrastructures and the number of vehicles in the task offloading subcloudlet, respectively. A cloud-based mobile edge computing (MEC) offloading framework



(a) Number of PVs = 5

(b) Number of PVs = 10

Figure 6: Workload ratio of selected nodes with different number of PVs.

in vehicular networks was proposed, and an efficient computation offloading strategy based on contract theoretic approach was introduced to maximize the benefit of the MEC service provider and improve the utilities of the vehicles in [36]. Its computational complexity was given as $O(NM)$, where N is the number of computation tasks types, and M is the number of MEC servers. The computation complexity of the proposed offloading strategy is similar to that of the algorithms mentioned above.

7. Conclusion

In this paper, an offloading strategy in MPVEC is investigated to optimize the system performance with jointly considering the workload balance among computing nodes. First, a multiple offloading node selection algorithm is proposed to select appropriate PVs to take part in computing tasks. Furthermore, a workload allocation strategy based on the idea of dynamic game is presented to optimize system performance and consider the load balancing at the same time. Numerical results have demonstrated that the proposed offloading strategy in MPVEC can effectively reduce the system cost under delay constraint while achieving the load balancing of the system. In this work, only the computing resources of PVs are considered to optimize the system performance. In the future work, the communication resources allocation in multiple PV-assisted MEC will be researched. This study can be reviewed as a reference for task offloading in the vehicular network.

Data Availability

All the data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the 2020 State Grid Corporation of China Science and Technology Program under Grant 5700-202041398A-0-0-00.

References

[1] Y. Ku, D. Y. Lin, C. F. Lee et al., "5G radio access network design with the fog paradigm: confluence of communications and computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 46–52, 2017.

[2] Y. Shih, W. Chung, A. Pang, T. Chiu, and H. Wei, "Enabling low-latency applications in fog-radio access networks," *IEEE Network*, vol. 31, no. 1, pp. 52–58, 2017.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[4] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: mobility aware dynamic service placement for mobile edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 10, pp. 2333–2345, 2018.

[5] B. Mohammed, M. Hamdan, J. S. Bassi et al., "Edge computing intelligence using robust feature selection for network traffic classification in internet-of-things," *IEEE Access*, vol. 8, pp. 224059–224070, 2020.

[6] Y. R. B. al-Mayouf, N. F. Abdullah, O. A. Mahdi et al., "Real-time intersection-based segment aware routing algorithm for urban vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 7, pp. 2125–2141, 2018.

[7] J. Xu, L. Chen, and P. Zhou, "Joint service caching and task offloading for Mobile edge computing in dense networks," in *IEEE INFOCOM 2018- IEEE Conference on Computer Communications*, pp. 207–215, Honolulu, HI, USA, 2018.

[8] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical fog-cloud computing for IoT systems: a computation offloading game," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3246–3257, 2018.

[9] Y. Hung and C. Wang, "Fog micro service market: promoting fog computing using free market mechanism," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–6, Barcelona, Spain, 2018.

[10] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: a viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[11] Z. Wang, D. Zhao, M. Ni, L. Li, and C. Li, "Collaborative Mobile computation offloading to vehicle-based cloudlets," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 1, pp. 1–781, 2020.

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

[13] J. Sun, Q. Gu, T. Zheng, P. Dong, and Y. Qin, "Joint communication and computing resource allocation in vehicular edge computing," *International Journal of Distributed Sensor Networks*, vol. 15, no. 3, 2019.

[14] K. Nguyen, S. Drew, C. Huang, and J. Zhou, "EdgePV: collaborative edge computing framework for task offloading," in *ICC 2021 - IEEE International Conference on Communications*, Montreal, QC, Canada, 2021.

[15] W. Qi, Q. Li, Q. Song, L. Guo, and A. Jamalipour, "Extensive edge intelligence for future vehicular networks in 6G," *IEEE Wireless Communications*, vol. 28, no. 4, pp. 128–135, 2021.

[16] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked vehicle edge computing: exploiting opportunistic resources for distributed Mobile applications," *IEEE Access*, vol. 6, pp. 66649–66663, 2018.

[17] F. H. Rabman, A. Y. M. Iqbal, S. H. S. Newaz, A. T. Wan, and M. S. Ahsan, "Street parked vehicles based vehicular fog computing: Tcp throughput evaluation and future research direction," in *2019 21st International Conference on Advanced Communication Technology (ICACT)*, pp. 26–31, Pyeong-Chang, Korea (South), 2019.

[18] Y. R. B. al-Mayouf, M. Ismail, N. F. Abdullah et al., "Efficient and stable routing algorithm based on user mobility and node density in urban vehicular network," *PLoS One*, vol. 11, no. 11, pp. 1–24, 2016.

[19] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint offloading and resource allocation in vehicular edge computing and networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7, Abu Dhabi, United Arab Emirates, 2018.

[20] F. Lin, X. Lü, I. You, and X. Zhou, "A novel utility based resource management scheme in vehicular social edge computing," *IEEE Access*, vol. 6, pp. 66673–66684, 2018.

[21] Z. Haitao, D. Yi, Z. Mengkang, W. Qin, S. Xinyue, and Z. Hongbo, "Multipath transmission workload balancing optimization scheme based on mobile edge computing in vehicular heterogeneous network," *IEEE Access*, vol. 7, pp. 116047–116055, 2019.

[22] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[23] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[24] P. Wang, C. Yao, Z. Zheng, G. Sun, and L. Song, "Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2872–2884, 2019.

[25] 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)*, pp. 1–6, Paris, France, 2017.

[26] L. Zhang and N. Ansari, "Latency-aware IoT service provisioning in UAV-aided mobile-edge computing networks," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10573–10580, 2020.

[27] Y. Liu, S. Xie, and Y. Zhang, "Cooperative offloading and resource management for UAV-enabled Mobile edge computing in power IoT system," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 12229–12239, 2020.

[28] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 48–54, 2018.

[29] R. Yadav, W. Zhang, O. Kaiwartya, H. Song, and S. Yu, "Energy-latency tradeoff for dynamic computation offloading in vehicular fog computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 14198–14211, 2020.

[30] R. Yadav, W. Zhang, I. A. Elgendy et al., "Smart healthcare: RL-based task offloading scheme for edge-enable sensor networks," *IEEE Sensors Journal*, vol. 21, no. 22, pp. 24910–24918, 2021.

[31] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, "PVA in VANETs: stopped cars are not silent," in *2011 Proceedings IEEE INFOCOM*, pp. 431–435, Shanghai, China, 2011.

[32] F. Malandrino, C. Casetti, C. Chiasserini, C. Sommer, and F. Dressler, "The role of parked cars in content downloading for vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 9, pp. 4606–4617, 2014.

[33] C. Ma, J. Zhu, M. Liu, H. Zhao, N. Liu, and X. Zou, "Parking edge computing: parked-vehicle-assisted task offloading for urban VANETs," *IEEE Internet of Things Journal*, vol. 8, no. 11, pp. 9344–9358, 2021.

[34] Y. Li, B. Yang, Z. Chen, C. Chen, and X. Guan, "A Contract-Stackelberg Offloading Incentive Mechanism for Vehicular Parked-Edge Computing Networks," in *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pp. 1–5, Kuala Lumpur, Malaysia, 2019.

[35] X. Huang, R. Yu, D. Ye, L. Shu, and S. Xie, "Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3773–3787, 2021.

[36] 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)*, pp. 288–294, Halmstad, Sweden, 2016.