





Research Article

A Game-Theoretic Scheme for Parked Vehicle-Assisted MEC Computation Offloading

Ruoyu Chen ^{1,2,3} Yanfang Fan ^{1,2} Mengxin Jia ¹ and Shuang Yuan ¹

¹School of Computer Science, Beijing Information Science & Technology University, Beijing, China

²Beijing Key Laboratory of Internet Culture & Digital Dissemination Research,
Beijing Information Science & Technology University, Beijing, China

³Southeast Institute of Information Technology, Beijing Institute of Technology, Fujian 351100, China

Correspondence should be addressed to Yanfang Fan; fyfhappy@bistu.edu.cn

Received 2 April 2022; Revised 14 July 2022; Accepted 20 July 2022; Published 24 August 2022

Academic Editor: Xu Zhang

Copyright © 2022 Ruoyu Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

By offloading computation tasks, multi-access edge computing (MEC) supports diverse services and reduces delay and energy consumption of mobile devices (MDs). However, limited resources of edge servers may be the bottleneck for task computing in high-density scenarios. To address this challenge, by leveraging the underutilized resources of parked vehicles to execute tasks, we propose a parked vehicle-assisted multi-access edge computing (PV-assisted MEC) architecture, which enables MEC servers to expand their capability flexibly. To achieve efficient offloading, we propose a PV-assisted MEC offloading scheme in a multi-MD environment. We design a game-based distributed algorithm to minimize the overhead of MDs and further reduce the burden on the MEC server. Simulation results show that compared with the common MEC system, our scheme can reduce the burden on the MEC server by 5% and the offloading overhead by 17%.

1. Introduction

With the improvement of mobile devices' capabilities and the ever-increasing interest in mobile applications, delay-sensitive and computation-intensive mobile applications have been emerging and drawing significant attentions, spanning technologies such as augmented reality, speech-to-text conversion, image processing, and interactive online games. However, due to the scarcity of resources, mobile devices are usually unable to meet the massive computing demands. The solution to this problem lies in improving the communication infrastructure by computation offloading [1–3]. Multi-access edge computing (MEC) is regarded as a key technology and architectural concept for the improvement of the computation offloading efficiency. MEC aims at extending cloud computing capabilities to the edge. Mobile devices (MDs) can offload tasks to nearby network edge servers [4]. For example, video streams and images collected through sensors or cameras mounted on the vehicles must be processed in real time to detect surrounding objects, recognize traffic lights, etc., to ensure the safety of autonomous

driving. However, vehicles do not have the capacity to process large amounts of images and videos instantly, so tasks are offloaded to edge server for processing, reducing the incidence of traffic accidents. Computation offloading technology in MEC not only overcomes the shortage of computing capabilities on mobile terminals but also avoids huge latency caused by transferring tasks to the cloud [5, 6]. However, existing MEC servers tend to have lightweight computing resources due to cost constraints, which means they are still not well equipped to handle the ever-growing task demands.

Scholars have studied the problem that it is difficult for a single VEC server to meet the strict latency requirements of MDs. The authors in [5] proposed a tiered offloading framework for edge computing, which utilizes nearby backup computing servers to make up for the insufficient MEC server resources. Guo and Liu [2] proposed a cloud-MEC collaborative computation offloading scheme with centralized cloud and multi-access edge computing over Fi-Wi network architecture. In addition, idle resources in unmanned aerial vehicles (UAVs) were used as a

supplement to the edge computing server to provide effective resource utilization and reliable service [7].

With the rapid development of the automotive industry, vehicles are equipped with an ever-increasing amount of communication and computing resources. Several works have focused on vehicle-assisted edge network to improve network service quality by leveraging idle resources in vehicles. In this network, idle resources are used to compute tasks to assist the edge network as vehicles with idle resources approaching vehicles carrying computation tasks. In daily life, 70% of personal vehicles are parked for an average of more than 20 hours per day [8]. These parked vehicles have a lot of idle computing, storage, and communication resources, as well as plenty of energy. Therefore, utilizing these idle resources is a promising way to improve network efficiency.

The use of parked vehicles to support network services has two advantages that cannot be ignored. On the one hand, parked vehicles are relatively stable in terms of communication. A moving vehicle may change its position frequently, which may cause the connection between the vehicle and the server to become unstable and affect the efficiency of task execution. In contrast, parked vehicles may remain stationary for long periods of time. On the other hand, parked vehicles involved in task offloading indirectly extend the service area of VEC. Outside the coverage of roadside units (RSUs), parked vehicles can serve as static nodes and service infrastructure, alleviating the shortage of edge server resources and supporting interconnection between vehicles and servers [9].

In this work, unlike existing computation offloading studies, we focus on reducing MDs' delay and energy consumption to improve quality of service (QoS). In addition, parked vehicles that can be used as service nodes in this work include not only those parked centrally in parking lots but also those parked scattered on the roadside, where legally permitted. We focus on the design of parked vehicle-assisted MEC architecture and the corresponding efficient computation offloading scheme. The main contributions of this study are as follows:

- (i) A parked vehicle-assisted multi-access edge computing (PV-assisted MEC) architecture is presented, in which nearby parked vehicles can help extend the service capabilities of the MEC system.
- (ii) The offloading decision problem is formulated as a noncooperation game. A game-based PV-assisted task offloading algorithm (GPTOA) is proposed, which decides whether each MD should offload and, if so, to which channel of MEC server or which PV.
- (iii) Simulation results show that the GPTOA not only effectively reduces the burden on the MEC server but also achieves significant performance improvements in terms of offloading overhead.

The rest of this study is organized as follows. First, related works are discussed in Section 2. Second, the PV-assisted MEC architecture is described in Section 3. Next, Section 4 presents the system model. After that, Section 5

formulates the task offloading problem and proposes a game-based PV-assisted task offloading algorithm. Extensive simulation results are provided in Section 6, followed by conclusions in Section 7.

2. Related Work

There are a number of studies focusing on mobile applications in MEC. Most of these focused on processing data and improving service qualities [10–15]. Zhang et al. [16] considered load balancing of computation resources on the edge servers and the highly dynamical nature of the vehicular networks, which led them to introduce fiber-wireless (Fi-Wi) technology to enhance vehicle edge computing network (VECN). Then, they used a game theory-based nearest task offloading algorithm and an approximate load balancing task offloading algorithm to solve the delay minimization problem. Cheng et al. [17] proposed a method to predict Wi-Fi offload potential and access costs by jointly considering user satisfaction, offload performance, and mobile network operators' revenues. The results showed that this scheme can improve the average utility of users and reduce service latency. Chen et al. [18] showed that it is NP-hard to find centralized optimum for task offloading in MEC with the goal of minimizing the overall computation overhead. Hence, they adopt a game-theoretic approach for achieving efficient offloading in a distributed manner.

The recent advent of vehicle-to-everything (V2X) communication technology makes vehicles an important network resource for improving network performance. Ding et al. [19] used CR (cognitive radio) router-enabled vehicles to transmit data to the desired location. Feng [20] proposed the hybrid vehicle edge cloud (HVC) framework, which made it possible to share available resources with neighboring vehicles through vehicle-to-vehicle (V2V) communication. Zhang et al. [21] investigated the effectiveness of computational transport strategies for vehicle-to-infrastructure (V2I) and V2V communication modes. They proposed an efficient predictive combination-mode relegation scheme that adaptively offloaded tasks to the MEC servers via direct uploading or predictive relay transmissions. Huang et al. [22] introduced the concept of vehicle neighbor group (VNG), which made it convenient to share similar services through V2V communication. Considering the similarity of tasks and computational capability of vehicles, Qiao et al. [23] divided vehicles into task computing sub-cloudlet and task offloading sub-cloudlet. Based on the two sub-cloudlets, they proposed a collaborative task offloading scheme that can effectively reduce the number of similar tasks transferred to MEC servers.

Furthermore, certain existing works focused on exploring ways to leverage the communication, storage, and computation capacity of parked vehicles, in which vehicles became service nodes for computation offloading. Liu et al. [24] proposed a vehicle edge computing network architecture in which vehicles act as edge servers to compute tasks. A problem with the objective of maximizing the long-term utility of the VEC network was presented in the study, modeled as a Markov decision process, and solved using two

reinforcement learning methods. Huang et al. [25] modeled the relationship between users, MEC server, and parking lot as a Stackelberg game. They presented a sub-gradient-based iterative algorithm to determine the workload distribution among parked vehicles and minimize the overall cost to the users. Li et al. [26] proposed a three-stage contract-Stackelberg offloading incentive mechanism to maximize the utility of vehicles, operators, and parking lot agents. Han et al. [27] proposed a dynamic pricing strategy that minimizes the average cost of the MEC system under the constraints on service quality by continuously adjusting the price according to the current system state.

By introducing parking lots as agents, many existing studies focused on utilizing the communication and computation capabilities of parked vehicles. The benefits and costs of parking vehicles and the costs to service users were taken into account. However, in addition to the vehicles parked centrally in parking lots, computing and communication channel resources of vehicles scattered on the roadside are not negligible. Moreover, in most cases, the quality of user experience should be prioritized. Therefore, in this study, based on the research work proposed in [18], we propose an PV-assisted MEC architecture to enhance the MEC network, in which parked vehicles can serve MDs directly. In addition, we propose a game-based task offloading algorithm to minimize the delay and energy consumption for service users.

3. Parked Vehicle-Assisted Multi-Access Edge Computing Architecture

With the advent of smart cars, more and more cars can be awakened to perform tasks even when parked. For example, when a parked Tesla car is in sentry mode or dog mode, some of its safety-related features are still working. With the increasing development of artificial intelligence, we believe that cars will become increasingly more intelligent. In the future, parked cars may support some modes that could provide services to other vehicles. The research presented in this study is conducted on this premise.

Although aspects such as incentives, communication costs, security, and scheduling should be considered if onboard computers in parked vehicles are to be used for edge computing, the focus of this study was on computation offloading strategy. Therefore, these aspects are not considered in this study, but should be taken into consideration in future studies to pursue a more complete solution.

A representative PV-assisted MEC service scenario is illustrated in Figure 1. There are a large number of MDs and parked vehicles running computationally intensive and delay-sensitive mobile applications. However, lightweight MEC servers and limited bandwidth resources are insufficient for these applications. Idle resources in parked vehicles can be used to relieve the pressure on the MEC. However, due to “selfishness,” not all parked vehicles are willing to provide resources. We assume that some parked vehicles can be recruited through certain incentives, such as extended parking opportunities or reduced parking fees. In addition, we assume that the MEC system can certify recruited parked

vehicles to ensure the security of the service and can update and monitor available resources of these parked vehicles in real time to improve resource utilization. We refer to these recruited certified parked vehicles as PVs. In summary, both MEC servers and PVs can provide services to MDs.

Figure 2 illustrates a representative PV-assisted MEC network architecture. Based on the original vehicular edge computing architecture, we move the vehicles capable of providing services from the device layer to the MEC layer to enable utilization of parked vehicles’ resources and allow them to provide services directly to MDs.

- (1) Cloud Layer. The first layer provides centralized cloud computing services and management functions such as critical or complex event handling, key data backup, and information authentication. The PV-assisted MEC architecture employs a software-defined network (SDN) controller to program, manipulate, and configure network in a logically centralized way.
- (2) Edge Cloud Layer (MEC Layer). The second layer consists of edge network access devices (e.g., RSU and base station (BS)) and data service devices (e.g., MEC servers and PVs). Edge network access devices are used for communication among edge facilities or between layers. MEC servers with lightweight storage and computing capabilities are deployed on edge network access devices. MEC servers are responsible for collecting service status information from themselves and from PV service nodes parked in the coverage area of RSU. Based on this information, MEC servers can process or assign tasks to MDs. By moving PVs from mobile device layer to MEC layer, the service capacity can be improved and bandwidth consumption can be reduced.
- (3) Mobile Device Layer. The third layer consists of mobile devices requesting services, such as vehicles, smartphones, tablets, and laptops. MDs request services by connecting to BSs via cellular network. MEC server and parked vehicles can provide services to terminal devices via cellular network or V2X. Here, V2X may be a link via cellular network or a link via dedicated short-range communications (DSRCs). Note that as a special kind of mobile device, vehicles are divided into two categories in this study: PVs and others. The former are located at the MEC layer as service providers, while the latter are located at the mobile device layer as service requesters.

Figure 3 illustrates the communication procedure between MD, MEC server, and PVs. First, when an MD generates a task, it sends the task request to the MEC server. Second, through iterative negotiation between the MEC server and the MDs, the task allocation result is calculated based on the status of the MDs and the MEC server. Then, the MEC server returns the task allocation result to the MD. When the task allocation result indicates that the task should be offloaded to a PV, the MEC also needs to notify the relevant PV (dotted arrow). Third, the MD sends task input

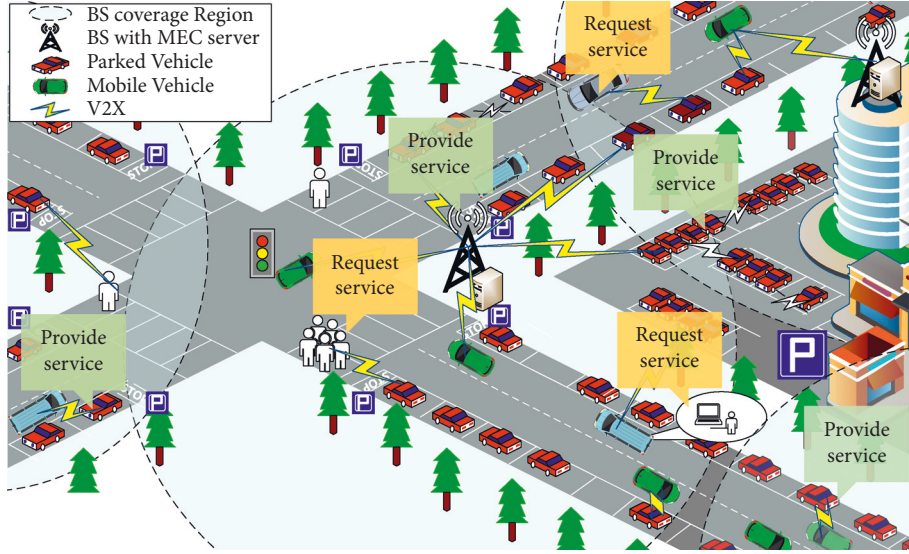


FIGURE 1: PV-assisted MEC service scenario.

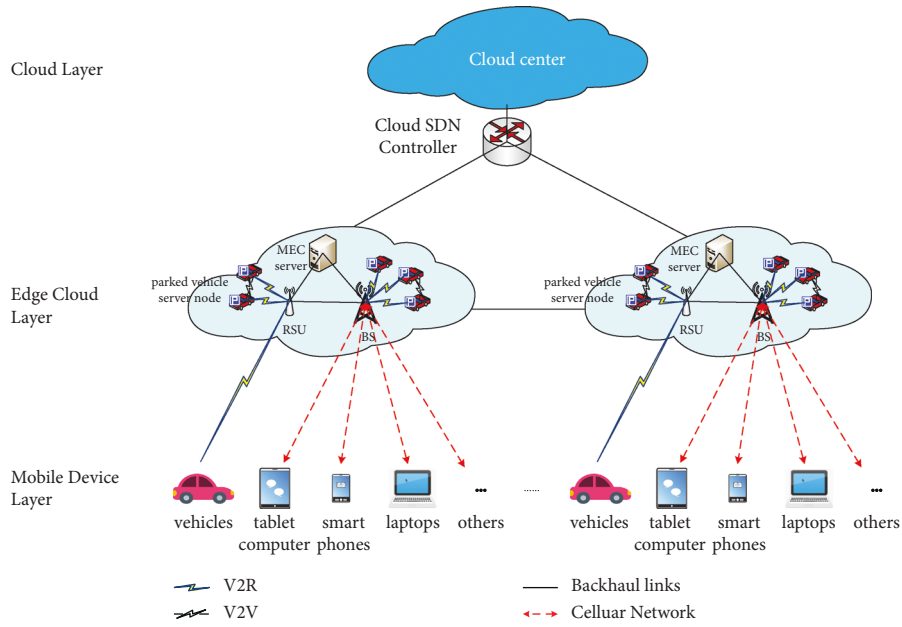


FIGURE 2: PV-assisted MEC network architecture.

data to the MEC server (solid line) or PV (dotted line) according to the task allocation information. Fourth, the MEC server (solid line) or PV (dotted line) processes the task and then returns the task result to the MD. Finally, the MD obtains the result and sends service satisfaction information back to the MEC server to reward the specific PV.

4. System Model

4.1. Network Model. We assign a unique identifier to each task and record the characteristics of tasks, such as traffic size and computation workload, in a globally shared feature table: $\mathcal{T} = \{T_1, \dots, T_M\}$. Without loss of generality, we assume that each MD generates only one task $T_i \triangleq \{d_i, c_i\}$,

$i \in \mathcal{M} = \{1, 2, \dots, M\}$, in a time period and tasks cannot be further divided. Here, d_i denotes the size of the task generated by MD i and c_i denotes the computation resources required by this task. We assume the existence of a wireless BS through which any MD can offload its computation task to a nearby MEC server (MS). Each wireless BS has C orthogonal frequency channels, denoted as $\mathcal{C} = \{1, 2, \dots, C\}$. Besides, in the coverage area of a BS, there is a set of PVs, denoted by $P = \{C + 1, \dots, C + P\}$. We consider a quasi-static scenario where the status of MDs, PVs, channels, and the MEC server remains unchanged for a given time period, whereas in different time periods, the status may change. For simplicity, we ignore the cost of establishing secure connections during transmissions. We denote $s_i \in \{0\} \cup C \cup P$,

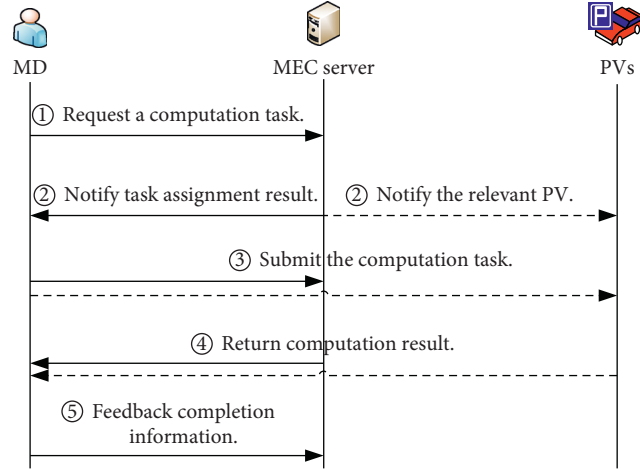


FIGURE 3: Sequence diagram for communication procedure.

$i \in \mathcal{M}$, as the selection decision variable. As shown in (1), let $s_i = 0$ denote that MD i executes its task locally, and $s_i > 0$ denotes that MD i chooses to offload this task. When $s_i = j$, $j \in \mathcal{C}$ indicates that MD i will offload task T_i to MEC server via channel j , while $j \in \mathcal{P}$ indicates that the task will be executed by PV j . Let $\mathbf{s} = \{s_1, s_2, \dots, s_M\}$ denote the set of selection decisions for all MDs. For ease of reference, we list key notations used in this study in Table 1.

$$s_i = \begin{cases} 0, & \text{if MD } i \text{ computes task } T_i \text{ locally,} \\ j, j \in \mathcal{C}, & \text{if MD } i \text{ off load task } T_i \text{ to MEC server via channel } j, \\ k, k \in \mathcal{P}, & \text{if MD } i \text{ off load task } T_i \text{ to PV } k. \end{cases} \quad (1)$$

4.2. Communication Model. In this section, we try to define the transmission rate of offloading. It is assumed that mobile device is equipped with a single antenna that can transmit data for one task at a time. When many MDs offload their tasks to the same MEC server, severe wireless channel interference may occur. Therefore, wireless channel conditions should be considered during transmission. If MD i chooses to offload its task to the MS via wireless channel, the data transmission rate for T_i can be expressed as follows:

$$r_i^{MS} = W_M \log_2 \left(1 + \frac{q_i h_i^{MS}}{\omega_0 + \sum_{k \in \mathcal{M}/\{i\}: s_j = s_k \in \mathcal{C}} q_k h_k^{MS}} \right). \quad (2)$$

Here, W_M is the bandwidth, q_i and h_i^{MS} are the transmission power and channel gain of MD i to the MS via nearby BS, respectively, and ω_0 is the background noise; $\sum_{k \in \mathcal{M}/\{i\}: s_j = s_k \in \mathcal{C}} q_k h_k^{MS}$ is the wireless channel interference generated by other MDs using the same channel.

MD i and PV j can communicate with each other only if the distance between them is less than a certain distance d_{\max}^{V2V} . We assume that any PV can only serve one MD during the computation offloading period. Therefore, there are no channel conflicts between MDs when tasks are offloaded to PVs. When MD i offloads its task T_i to PV j that is not

occupied by other MDs, the data transmission rate can be expressed as follows:

$$r_i^{PV} = W_P \log_2 \left(1 + \frac{q_i h_i^{PV}}{\omega_0} \right). \quad (3)$$

Here, W_P is the bandwidth between MD and PV.

4.3. Computation Model of Mobile Devices. We use f_i^{Loc} to denote the computational power of MD i . Thus, the delay of the locally executed task T_i can be expressed as follows:

$$t_i^{Loc} = \frac{c_i}{f_i^{Loc}}. \quad (4)$$

Similar to [28], we assume that the power consumption of a certain MD is proportional to the cube of its computational power. The energy consumption coefficient μ is related to the chip's hardware architecture. The device's energy consumption for local execution can be expressed as follows:

$$e_i^{Loc} = \mu t_i^{Loc} (f_i^{Loc})^3. \quad (5)$$

Considering that MDs are usually energy and delay sensitive, we define parameters α_i and β_i ($\alpha_i, \beta_i \in [0, 1]$, $\alpha_i + \beta_i = 1$) as the weights for delay and energy in the computing of overhead for MD i , respectively. MDs tend to save time (larger α_i) when tasks are delay sensitive, and they tend to save energy (larger β_i) when batteries are low.

Thus, the overhead of local execution can be expressed as follows:

$$K_i^{Loc} = \alpha_i t_i^{Loc} + \beta_i e_i^{Loc}. \quad (6)$$

4.4. Computation Model of MEC Server. For most mobile applications, such as fingerprint, face, or iris recognition, and sensor data processing, the size of the computation result is much smaller than the size of the input data. We ignore the transmission time of computation results.

TABLE 1: Notations.

Symbol	Description
$M = \{1, 2, \dots, M\}$	Set of mobile devices
$C = \{1, 2, \dots, C\}$	Set of channels on the MEC server
$P = \{C + 1, \dots, C + P\}$	Set of parked vehicles
$\mathcal{T} = \{T_1, \dots, T_M\}$	Set of tasks generated by MD
$\mathbf{s} = \{s_1, \dots, s_M\}$	Selection decisions set of tasks
α_i, β_i	Delay and energy weights, $\alpha_i, \beta_i \in [0, 1], \alpha_i + \beta_i = 1$
d_i	Computation data size of task T_i (bit)
c_i	Computing resources required by task T_i (cycles)
$f_i^{Loc}, f_i^{MS}, f_i^{PV}$	Computing ability of MD, MS, or PV i (Hz)
μ	Energy consumption coefficient
$d_{i,PV_j}^{V2V}, d_{i,MS}^{V2I}$	The distance between MD i and PV j / MS (m)
$d_{\max}^{V2V}, d_{\max}^{V2I}$	The maximum communication distance of V2V/V2I (m)
γ_i	Received interferences of MD i
ω_0	Background noise (dBm)
W_M, W_P	Uplink channel bandwidth between MD and MS/PV (Hz)
q_i	Transmission power of MD i (W)
h_i^{MS}, h_i^{PV}	Channel gain from MD i to the MS/PV
r_i^{MS}, r_i^{PV}	Transmission rate of MD i to MS/PV (bps)
$t_i^{Loc}, t_i^{MS}, t_i^{PV}$	Delay for local execution, offloading to MS/PV (seconds)
$e_i^{Loc}, e_i^{MS}, e_i^{PV}$	Energy consumption for local execution, offloading to MS/PV (joules)
$K_i^{Loc}, K_i^{MS}, K_i^{PV}$	Total overhead for local execution, offloading to MS/PV

Therefore, the delay for offloading task T_i to MEC server MS can be divided into two parts: data uploading time and task execution time, expressed as follows:

$$t_i^{MS} = t_{i,up}^{MS} + t_{i,exe}^{MS} = \frac{d_i}{r_i^{MS}} + \frac{c_i}{f_i^{MS}}, \quad (7)$$

where f_i^{MS} is MS' computing capability.

Usually, the MEC server has sufficient power supply, so the energy consumption on the MEC server can be ignored. From MD's perspective, the energy consumption of offloading task to MS comes from transmitting data over wireless network and can be expressed as follows:

$$e_i^{MS} = \frac{q_i d_i}{r_i^{MS}}. \quad (8)$$

Thus, the overhead for offloading task T_i to MEC server can be expressed as follows:

$$K_i^{MS} = \alpha_i t_i^{MS} + \beta_i e_i^{MS}. \quad (9)$$

4.5. Computation Model of Parked Vehicles. Let f_j^{PV} denote the computing resource allocated to task T_i from PV j . The delay for offloading task T_i to PV j ($j \in P$) can be expressed as follows:

$$t_i^{PV} = \frac{d_i}{r_i^{PV}} + \frac{c_i}{f_j^{PV}}. \quad (10)$$

Similarly, energy consumption on PV j is ignored (which will be considered in future works), and energy consumption on MD for offloading task T_i to PV j can be expressed as follows:

$$e_i^{PV} = \frac{q_i d_i}{r_i^{PV}}. \quad (11)$$

Thus, the overhead of MD i for offloading task T_i to PV j can be expressed as follows:

$$K_i^{PV} = \alpha_i t_i^{PV} + \beta_i e_i^{PV}. \quad (12)$$

5. Problem Formulation and Algorithm Design

5.1. Problem Formulation. According to Section 4, the overhead of task T_i can be expressed as follows:

$$K_i(s_i) = \begin{cases} K_i^{Loc}, & \text{if } s_i = 0, \\ K_i^{MS}, & \text{if } s_i = j, j \in C, \\ K_i^{PV}, & \text{if } s_i = k, k \in P. \end{cases} \quad (13)$$

There are $1 + C + P$ choices available for each task. Delay and energy consumption may vary depending on offloading strategies. Therefore, the overall goal is to minimize the total overhead of all MDs. Thus, the problem of optimizing the total overhead for all MDs can be expressed as follows:

$$\begin{aligned} & \min_s \sum_{i=1}^M K_i(s_i), \\ & \text{s.t. } \sum_{i \in M} I_{\{s_i=0\}} + \sum_{i \in M} I_{\{s_i \in C\}} + \sum_{i \in M} I_{\{s_i \in P\}} = M, \\ & \sum_{i=1}^M I_{\{s_i=j\}} \leq 1, \quad \forall j \in P, \\ & d_{i,PV_j}^{V2V} \leq d_{\max}^{V2V}, \quad \forall s_i = j, j \in P, \\ & d_{i,MS}^{V2I} \leq d_{\max}^{V2I}, \quad \forall s_i = j, j \in C. \end{aligned} \quad (14)$$

Here, $I_{\{E\}}$ is the indicator function with $I_{\{E\}} = 1$ if the event E is true and $I_{\{E\}} = 0$ otherwise. There are four constraints for problem (P). Constraint (C1) is that every task should be executed. Constraint (C2) is that each PV serves at most one MD. Constraint (C3) is that MD i and PV j can communicate only when they are close enough to each other. Similarly, constraint (C4) is that MD i and the MEC server can communicate only when they are close enough to each other.

The task set \mathcal{T} can be divided into three mutually exclusive subsets by the selection decisions: $\mathcal{T} = \mathcal{T}_{Loc} \cup \mathcal{T}_{MS} \cup \mathcal{T}_{PV}$. \mathcal{T}_{Loc} means that tasks are processed locally, \mathcal{T}_{MS} means that tasks are offloaded to the MS, and \mathcal{T}_{PV} means that tasks are offloaded to some PV.

By incorporating PVs as extra service providers for computation offloading, the problem proposed in this study is essentially a generalization of that proposed in [18]. However, it has been shown in [18] that the centralized optimization problem for minimizing the system-wide computation overhead is NP-hard. Therefore, with PVs as additional computation offloading providers, the problem proposed in this study is also NP-hard and difficult to solve. Similar to [18], the centralized cost minimizing problem for PV-assisted MEC computation offloading can be transformed into a distributed computation offloading decision problem among mobile device users. In the computation offloading process, each MD wants to reduce its overhead as much as possible. Therefore, they need to be aware of the choices made by other MDs. Let $s_{-i} = (s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_M)$ be the selection decisions by all other MDs except MD i . Based on s_{-i} , MD i can make a proper decision s_i to reduce its overhead. The distributed computation offloading problem ((P')) can be defined as follows:

$$\min_{s_i \in \mathcal{S}} K_i(s_i, s_{-i}), \quad \forall i \in \mathcal{M}, \quad (15)$$

in which the overhead function of mobile device i can be defined as follows:

$$K_i(s_i, s_{-i}) = \begin{cases} K_i^{Loc}, & \text{if } s_i = 0, \\ K_i^{MS}, & \text{if } s_i = j, 0 < j \leq C, \\ K_i^{PV}, & \text{if } s_i = j, C < j \leq C + P, \quad \forall k \in \mathcal{M} \setminus \{i\}, s_i \neq s_k. \end{cases} \quad (16)$$

Problem p' can be formulated as a noncooperative game: $G = (\mathcal{M}, \{\mathcal{S}_i\}_{i \in \mathcal{M}}, \{K_i\}_{i \in \mathcal{M}})$ with finite players, where \mathcal{M} is the set of players, \mathcal{S}_i is the set of selection decisions for player/MD i , and the overhead function $K_i(s_i, s_{-i})$ is the cost function to be minimized by each MD i .

In the next subsection, we will analyze the existence of Nash equilibrium in the PV-assisted MEC computation offloading game.

5.2. Nash Equilibrium Analysis. Here is the definition of the important concept of Nash equilibrium [29].

Definition 1. A selection decision set $s^* = (s_1^*, \dots, s_M^*)$ is a Nash equilibrium of the PV-assisted MEC computation offloading game, if at the equilibrium s^* , no MD can further reduce its overhead by unilaterally changing its selection decision, i.e.,

$$K_i(s_i^*, s_{-i}^*) \leq K_i(s_i, s_{-i}^*), \quad \forall s_i \in \mathcal{S}_i, i \in \mathcal{M}. \quad (17)$$

To study the existence of Nash equilibrium, we will first introduce the concept of potential game [30].

Definition 2. A game is said to be an ordinal potential game if the incentive of all players to change their strategy can be expressed using a single global function called the potential function: $\Phi(\mathbf{s})$, such that $\forall i \in \mathcal{M}, s_{-i} \in \prod_{k \neq i} \mathcal{S}_k$, and $s'_i, s_i \in \mathcal{S}_i$, if

$$K_i(s'_i, s_{-i}) < K_i(s_i, s_{-i}). \quad (18)$$

Then,

$$\Phi(s'_i, s_{-i}) < \Phi(s_i, s_{-i}). \quad (19)$$

An important feature of the finite ordinal potential game is that it always has a Nash equilibrium and it has the finite improvement property. In other words, if finite players start with an arbitrary strategy profile and iteratively deviate to their unique best replies in each period, the process terminates in an NE after finite steps. Next, before giving detailed proof that the PV-assisted MEC computation offloading game is an ordinal potential game, we have the following lemma.

Lemma 1. *Given a strategy profile \mathbf{s} , MD i can reduce its computation overhead by offloading its task to the MEC server if condition (C_m) holds and offloading its task to PV if condition (C_p) holds.*

(C_m): the received interference $\gamma_i(\mathbf{s}) = \sum_{k \in \mathcal{M} \setminus \{i\}} s_k = s_k \in \mathcal{E} q_k h_k^{MS}$ satisfies $\gamma_i(\mathbf{s}) < T_i^{ML}$ and $\gamma_i(\mathbf{s}) < T_i^{MP}$, with the following thresholds:

$$T_i^{ML} = \frac{q_i h_i^{MS}}{2(\alpha_i d_i + \beta_i q_i d_i / W_M (K_i^{Loc} - \alpha_i t_{i,exe}^{MS})) - 1} - \omega_0, \quad (20)$$

$$T_i^{MP} = \frac{q_i h_i^{MS}}{2(\alpha_i d_i + \beta_i q_i d_i / W_M (K_i^{PV} - \alpha_i t_{i,exe}^{MS})) - 1} - \omega_0. \quad (21)$$

(C_p): PV j is not occupied by any other MD; i.e., $j \in P, \sum_{k \in \mathcal{M} \setminus \{i\}} I_{\{s_k=j\}} = 0$, and the computation overhead K_i^{PV}, K_i^{Loc} satisfies $K_i^{PV} < K_i^{Loc}$, and

$$\gamma_i(\mathbf{s}) > T_i^{MP}. \quad (22)$$

Proof. For condition C_m: according to equation (14), we know that when the overhead satisfies $K_i^{MS} < \min\{K_i^{Loc}, K_i^{PV}\}$, i.e., $K_i^{MS} < K_i^{Loc}$ and $K_i^{MS} < K_i^{PV}$, the best strategy for MD i is to offload its task to the MEC server.

According to equations (7) to (9), the condition $K_i^{MS} < K_i^{Loc}$ is equivalent to the following:

$$\alpha_i \frac{d_i}{r_i^{MS}} + \alpha_i t_{i,exe}^{MS} + \beta_i \frac{q_i d_i}{r_i^{MS}} < K_i^{Loc}. \quad (23)$$

That is,

$$r_i^{MS} > \frac{\alpha_i d_i + \beta_i q_i d_i}{K_i^{Loc} - \alpha_i t_{i,exe}^{MS}}. \quad (24)$$

According to (2), we then have the following:

$$\sum_{k \in \mathcal{M}/\{i\}: s_i = s_k \in \mathcal{C}} q_k h_k^{MS} < \frac{q_i h_i^{MS}}{2^{\alpha_i d_i + \beta_i q_i d_i / W_M} (K_i^{Loc} - \alpha_i t_{i,exe}^{MS}) - 1} - \omega_0, \quad (25)$$

which is $\gamma_i(\mathbf{s}) < T_i^{ML}$ in condition (C m).

According to equations (7) to (12), the condition $K_i^{MS} < K_i^{PV}$ is equivalent to the following:

$$\alpha_i \frac{d_i}{r_i^{MS}} + \alpha_i t_{i,exe}^{MS} + \beta_i \frac{q_i d_i}{r_i^{MS}} < K_i^{PV}. \quad (26)$$

Then, we have the following:

$$r_i^{MS} > \frac{\alpha_i d_i + \beta_i q_i d_i}{K_i^{PV} - \alpha_i t_{i,exe}^{MS}}. \quad (27)$$

Furthermore, according to (2), we can get $\gamma_i(\mathbf{s}) < T_i^{MP}$ in condition (C m).

For condition C p: the proof is straightforward and is omitted here.

Based on Lemma 1, we will show that the PV-assisted MEC computation offloading game is a potential game with the potential function as follows:

$$\begin{aligned} \Phi(\mathbf{s}) &= \frac{1}{2} \sum_{i=1}^M \sum_{k \neq i} q_i h_i^{MS} q_k h_k^{MS} I_{\{s_i = s_k\}} I_{\{s_i \in [1, C]\}} \\ &+ \sum_{i=1}^M q_i h_i^{MS} T_i^{MP} I_{\{s_i \in (C, C+P)\}} + \sum_{i=1}^M q_i h_i^{MS} T_i^{ML} I_{\{s_i = 0\}}. \end{aligned} \quad (28)$$

□

Theorem 1. *The PV-assisted MEC computation offloading game is a potential game with $\Phi(\mathbf{s})$ (equation 21) as the potential function and hence always has a Nash equilibrium and the finite improvement property.*

Proof. Suppose that MD $i \in \mathcal{M}$ updates its decision selection from s_i to s'_i , and this leads to a decrease in the overhead function, i.e., $K_i(s_i, s_{-i}) > K_i(s'_i, s_{-i})$. According to Definition 2, we must show that this also leads to a decrease in the potential function, i.e., $\Phi(s_i, s_{-i}) > \Phi(s'_i, s_{-i})$. There are eight possible cases.

- (1) $s_i \in [1, C]$ and $s'_i \in [1, C]$
- (2) $s_i \in (C, C+P]$ and $s'_i \in (C, C+P]$
- (3) $s_i \in [1, C]$ and $s'_i \in (C, C+P]$
- (4) $s_i \in (C, C+P]$ and $s'_i \in [1, C]$
- (5) $s_i \in [1, C]$ and $s'_i = 0$
- (6) $s_i = 0$ and $s'_i \in [1, C]$
- (7) $s_i \in (C, C+P]$ and $s'_i = 0$
- (8) $s_i = 0$ and $s'_i \in (C, C+P]$

For case (1), according to equations (7) to (9), the premise $K_i(s_i, s_{-i}) > K_i(s'_i, s_{-i})$ implies that $r_i^{MS} > r_i'^{MS}$. Then, since the function $W_m \log_2(x)$ increases monotonically with the variable x , according to equation (2), $r_i'^{MS} > r_i^{MS}$ implies that

$$\sum_{k \in \mathcal{M}/\{i\}: s_i = s_k \in C} q_k h_k^{MS} > \sum_{k \in \mathcal{M}/\{i\}: s'_i = s_k \in C} q_k h_k^{MS}. \quad (29)$$

Since $s'_i \in [1, C]$ and $s_i \in [1, C]$, according to (28) and (29), we have the following:

$$\begin{aligned} \Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) &= \frac{1}{2} \sum_{i=1}^M \sum_{k \neq i} q_i h_i^{MS} q_k h_k^{MS} I_{\{s_i = s_k\}} - \frac{1}{2} \sum_{i=1}^M \sum_{k \neq i} q_i h_i^{MS} q_k h_k^{MS} I_{\{s'_i = s_k\}} \\ &= \frac{1}{2} q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s'_i = s_k\}} + \frac{1}{2} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} q_i h_i^{MS} \\ &\quad - \frac{1}{2} q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} - \frac{1}{2} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_k = s'_i\}} q_i h_i^{MS} \\ &= q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} - q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s'_i = s_k\}} > 0. \end{aligned} \quad (30)$$

For case (2), the premise $K_i(s_i, s_{-i}) > K_i(s'_i, s_{-i})$ is equivalent to $K_i^{PV} > K_i'^{PV}$. According to the definition of T_i^{MP}

(equation 19), if $K_i^{PV} > K_i'^{PV}$, then $T_i^{MP} > T_i'^{MP}$. Then, we have the following:

$$\begin{aligned}\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) &= \sum_{i=1}^M q_i h_i^{MS} T_i^{MP} I_{\{s_i \in (C, C+P)\}} - \sum_{i=1}^M q_i h_i^{MS} T_i^{MP} I_{\{s'_i \in (C, C+P)\}} \\ &= q_i h_i^{MS} T_i^{MP} - q_i h_i^{MS} T_i'^{MP} > 0.\end{aligned}\quad (31)$$

For case (3), since $s_i \in [1, C]$ and $s'_i \in (C, C + P]$, from Lemma 1 (condition C p), we have $\gamma_i(\mathbf{s}) > T_i'^{MP}$. This implies that

$$\begin{aligned}\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) &= \frac{1}{2} \sum_{i=1}^M \sum_{k \neq i} q_i h_i^{MS} q_k h_k^{MS} I_{\{s_i = s_k\}} I_{\{s_i \in [1, C]\}} - q_i h_i^{MS} T_i'^{MP} \\ &= \frac{1}{2} q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} + \frac{1}{2} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_k = s_i\}} q_i h_i^{MS} - q_i h_i^{MS} T_i'^{MP} \\ &= q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} - q_i h_i^{MS} T_i'^{MP} > 0.\end{aligned}\quad (32)$$

Case (4) is the opposite of case (3), and its proof is omitted here.

For case (5), since $s_i \in [1, C]$ and $s'_i = 0$, it can be deduced from $K_i(s_i, s_{-i}) > K_i(s'_i, s_{-i})$ that $K_i^{MS} > K_i^{Loc}$. According to Lemma 1 (condition C m), we have $\gamma_i(\mathbf{s}) > T_i^{ML}$. Then,

$$\begin{aligned}\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) &= \frac{1}{2} \sum_{i=1}^M \sum_{k \neq i} q_i h_i^{MS} q_k h_k^{MS} I_{\{s_i = s_k\}} I_{\{s_i \in [1, C]\}} - q_i h_i^{MS} T_i^{ML} \\ &= \frac{1}{2} q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} + \frac{1}{2} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_k = s_i\}} q_i h_i^{MS} - q_i h_i^{MS} T_i^{ML} \\ &= q_i h_i^{MS} \sum_{k \neq i} q_k h_k^{MS} I_{\{s_i = s_k\}} - q_i h_i^{MS} T_i^{ML} > 0.\end{aligned}\quad (33)$$

Case (6) is the opposite of case (5), and thus, its proof is omitted here.

For case (7), $s_i \in (C, C + P]$ and $s'_i = 0$ imply that $K_i^{PV} > K_i^{Loc}$. Then, according to the definitions of T_i^{ML} (Eq. 18) and T_i^{MP} (equation 19), we have $T_i^{MP} > T_i^{ML}$. Therefore,

$$\begin{aligned}\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) &= \sum_{i=1}^M q_i h_i^{MS} T_i^{MP} I_{\{s_i \in (C, C+P)\}} - \sum_{i=1}^M q_i h_i^{MS} T_i^{ML} I_{\{s_i = 0\}} \\ &= q_i h_i^{MS} T_i^{MP} - q_i h_i^{MS} T_i^{ML} > 0.\end{aligned}\quad (34)$$

Case (8) is the opposite of case (7), and thus, its proof is omitted here.

Combining results from the above cases, we can conclude that the PV-assisted MEC computation offloading game is a potential game. \square

```

(1) initialize: MDs' strategy set  $\mathbf{s}(0) = \{0, 0, \dots, 0\}$ ;
(2) repeat
(3)   for each MD  $i \in \mathcal{M}$  and each iteration  $t$ , calculate the overhead according to (16);
(4)   MD  $i$  makes the best response  $s'_i(t)$  as the selected strategy;
(5)   if  $s'_i(t) \neq s_i(t-1)$  then
(6)     send update request  $s'_i(t)$  to the MEC server;
(7)     if received  $s'_i(t)$  from MEC server then
(8)       set decision selection  $s_i(t) = s'_i(t)$ ;
(9)     else
(10)      keep the decision selection unchanged, i.e.,  $s_i(t) = s_i(t-1)$ ;
(11)    end if
(12)  else
(13)    keep the decision selection unchanged, i.e.,  $s_i(t) = s_i(t-1)$ ;
(14)  end if
(15) until received END message

```

ALGORITHM 1: Game-based PV-assisted task offloading algorithm (GPTOA).

5.3. Algorithm Design. Algorithm 1 illustrates the game-based PV-assisted task offloading algorithm (GPTOA) for problem (P') . Similar to [18], the algorithm will run iteratively on each MD. The main idea of GPTOA is that, based on the current state, each MD makes the best decision by calculating the overhead according to (16) (Line 3). Meanwhile, constraints (C1)–(C4) will be checked in each iteration. When constraints (C3) and (C4) cannot be satisfied, we set the overhead to infinity. During each iteration t , MD i updates its decision selection $s'_i(t)$ based on the best response and sends it to the MEC server as an update request, if $s'_i(t) \neq s_i(t-1)$. The MEC server randomly selects one decision selection $s'_i(t)$ from all update requests and sends $s'_i(t)$ back to MD i for updating its decision for the next iteration (Lines 4–8). The iteration continues until the decision selection remains unchanged. At the end, the MEC server will broadcast end message to all MDs and each MD will execute the computation task according to the last decision selection. According to the finite improvement property of potential game (Theorem 1), the algorithm will converge to a Nash equilibrium within finite number of iterations.

In GPTOA, MDs execute operations in parallel in each time slot. The most time-consuming operation is the computing of the best response update process in Line 3, which mainly involves the sorting operation over the overhead of available offloading strategies for all MDs. Since the sorting operation typically has a time complexity of $O(n \log n)$, and the maximum number of available choices for all MDs is not greater than $C + P + 1$, therefore, the computational complexity of each time slot will not exceed $O(x \log x)$, in which $x = C + P + 1$. If the algorithm takes y time slots to terminate, the total computational complexity of Algorithm 1 is $O(y \cdot x \log x)$.

Let $G_i = q_i h_i^{MS}$, $G_{\max} = \max_{i \in \mathcal{M}} \{G_i\}$, $G_{\min} = \min_{i \in \mathcal{M}} \{G_i\}$, and $T_{\max} = \max_{i \in \mathcal{M}} \{T_i^{ML}, T_i^{MP}\}$. For the upper bound of y , similar to [18], we have the following result.

Theorem 2. *When G_i and T_i are nonnegative integers for any $i \in \mathcal{M}$, the game-based PV-assisted task offloading algorithm will terminate within at most $(G_{\max}^2/2G_{\min})M^2 + (G_{\max}T_{\max}/G_{\min})M$ time slots, i.e., $y \leq (G_{\max}^2/2G_{\min})M^2 + (G_{\max}T_{\max}/G_{\min})M$.*

Proof. According to equation (21) and the definition of G_i , G_{\max} , G_{\min} , and T_{\max} , we have the following:

$$0 \leq \Phi(s) \leq \frac{1}{2} \sum_{i=1}^M \sum_{i=1}^M G_{\max}^2 + \sum_{i=1}^M G_{\max} T_{\max} = \frac{1}{2} G_{\max}^2 M^2 + G_{\max} T_{\max} M. \quad (35)$$

According to Theorem 1, during each time slot, MD $i \in \mathcal{M}$ updates its decision s_i to decision s'_i and this action leads to a decrease in its overhead function, i.e., $K_i(s_i, s_{-i}) > K_i(s'_i, s_{-i})$. The key idea of this proof is to show that this also leads to a decrease in the potential function by at least G_{\min} , i.e.,

$$\Phi(s_i, s_{-i}) \geq \Phi(s'_i, s_{-i}) + G_{\min}. \quad (36)$$

Similar to the proof of Theorem 1, there are eight cases to consider.

For case (1), $s_i \in [1, C]$ and $s'_i \in [1, C]$; according to (23), we have the following:

$$\Phi(s_i, s_{-i}) - \Phi(s'_i, s_{-i}) = G_i \left(\sum_{k \neq i} G_k I_{\{s_i = s_k\}} - \sum_{k \neq i} G_k I_{\{s'_i = s_k\}} \right) > 0. \quad (37)$$

Since G_i are nonnegative integers, we have the following:

$$\sum_{k \neq i} G_k I_{\{s_i = s_k\}} \geq \sum_{k \neq i} G_k I_{\{s'_i = s_k\}} + 1. \quad (38)$$

Then, based on (37):

$$\Phi(s_i, s_{-i}) \geq \Phi(s'_i, s_{-i}) + G_i \geq \Phi(s'_i, s_{-i}) + G_{\min}. \quad (39)$$

For other cases, the proofs are similar and are omitted here. \square

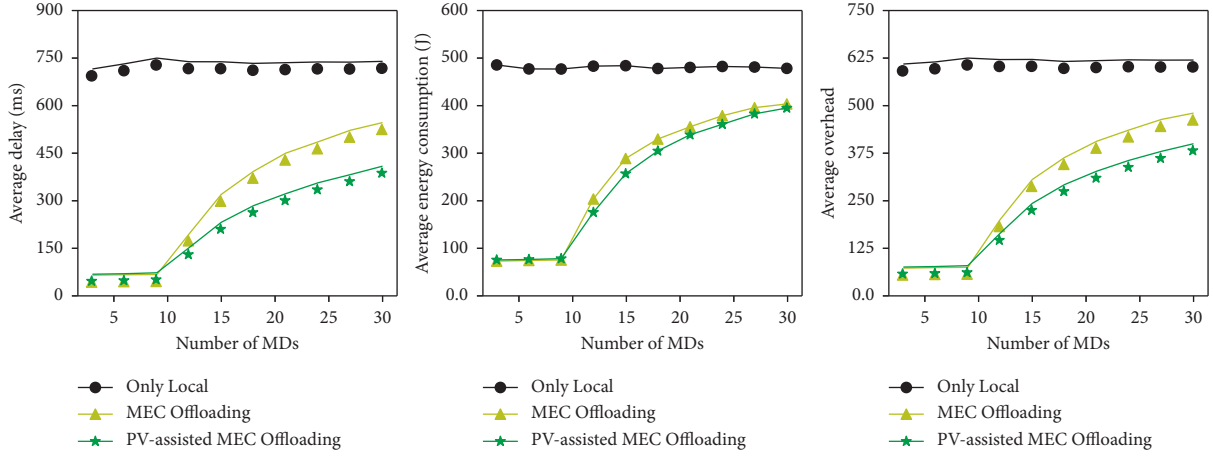


FIGURE 4: All MDs' average delay, energy consumption, and overhead with different numbers of MDs.

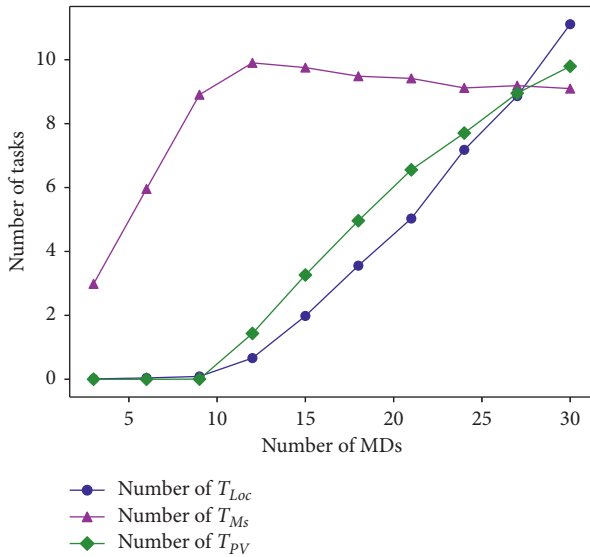


FIGURE 5: Results of task assignment with different numbers of MDs.

6. Simulation Results

6.1. Parameter Settings. The GPTOA was simulated and evaluated using Python with packages such as NumPy, random, and SciPy. We considered the scenario where the wireless BS had a coverage area of $50 * 50$ m². Each BS had $C = 10$ channels with a channel bandwidth of $W_M = 20$ MHz. The transmission power was $q_i = 400$ mWatts, and the background noise was $\omega_0 = -100$ dBm. Based on the radio interference model for urban cellular radio environment, we set the channel gain to $h_i^{MS} = d_{i,r}^{-\alpha}$, where $d_{i,r}$ was the distance between MD i and the wireless BS, and $\alpha = 4$ was the path loss factor [18]. The maximum communication distances of V2X were $d_{max}^{V2V} = 15$ m and $d_{max}^{V2I} = 200$ m, respectively [31]. The energy consumption coefficient was $\mu = 1$.

For computational tasks, the data size of offloaded task T_i was $d_i = 1$ MB. The total number of CPU cycles (c_i)

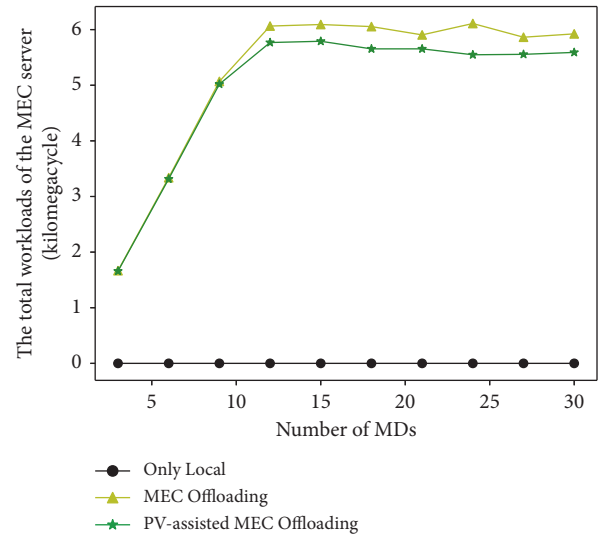


FIGURE 6: Total workload of the MEC server with different numbers of MDs.

required by task T_i was randomly distributed in the interval of $(1, 1000)$ megacycles. The weight parameters for all MDs were $\alpha_i = \beta_i = 0.5$. We assumed that the values of weight parameters remained constant during a single offloading process. Since most MEC servers are equipped with multiple CPUs, and multiple CPUs can be allocated to one MD at a time, for ease of computation, it was assumed that the computation power allocated to an MD by the MEC server was $f_i^{MS} = 10$ GHz. The computation power of MDs was randomly distributed between $[0.5, 1]$ GHz. The computation power of PVs was randomly distributed between $[0.5, 1, 1.5, 2]$ GHz. The communication bandwidth between PV and MD was $W_P = 20$ MHz.

6.2. Performance Analysis. To evaluate the scheme proposed in this work, we compared three schemes: (1) local only (scheme 1): all MDs decide to compute their own tasks

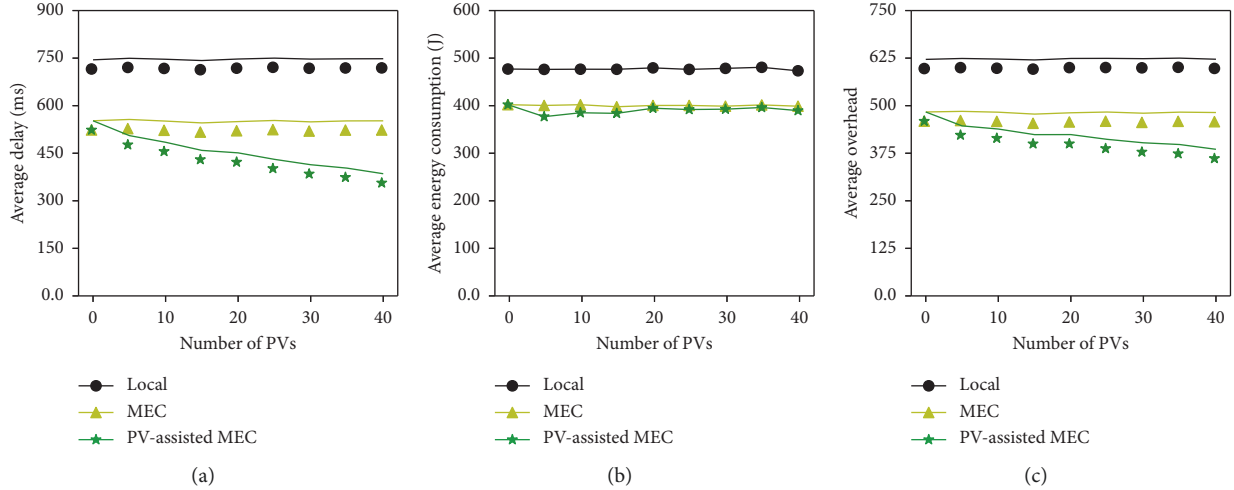


FIGURE 7: Average delay, energy consumption, and overhead of MDs with different numbers of PVs.

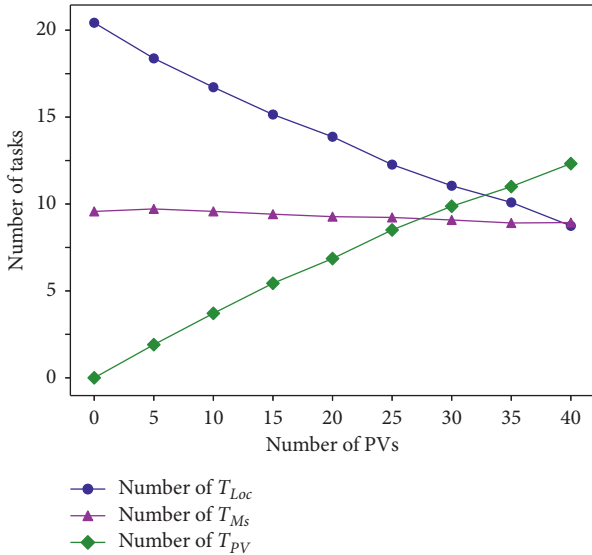


FIGURE 8: Task assignment of MDs with different numbers of PVs.

locally. (2) MEC offloading (scheme 2): the tasks are either computed locally or are offloaded to the MEC server [18]. (3) PV-assisted MEC offloading (our scheme): the tasks are computed locally, offloaded to the MEC or PVs. The work presented in [18] was treated as a special case with number of PVs set to 0 in this study. To eliminate the effect of randomness on the algorithm results, we conducted 1000 tests and performed statistical analysis of the results as follows.

First, we fixed the number of PVs (service vehicles) to 40 to observe the changes in the metrics (average delay, energy consumption, and total overhead of tasks, as well as task assignment results and load on the MEC server) as the number of MDs (service requesters) increased.

In Figure 4, the average delay, energy consumption, and total overhead of three schemes are compared. We can see that all three metrics of scheme 1 are higher than the other

schemes due to the limited local computation power. When the number of MDs is less than 10, the three metrics are the same for schemes 2 and 3. This can be explained by the lack of tasks offloaded to PVs. As the number of MDs increases, the metrics of scheme 3 grow less rapidly than the other two schemes. When the number of MDs is 30, scheme 3 results in a 26% reduction in delay and a 17% reduction in total overhead (on average) compared with scheme 2.

Figure 5 shows the task assignment results of the proposed scheme. When the number of MDs is less than 10, all tasks will be offloaded to MS, because MS has a shorter task execution time. However, as the number of MDs increases, no more tasks can be offloaded to MS. This is due to the fact that when multiple tasks are offloaded to MS, it leads to strong channel interference and heavier computational load, which in turn causes intolerable delays. This causes some MDs to give up offloading their tasks to the MS. In addition, due to limited resources and short communication distance, only a small portion of the PVs can serve MDs. Therefore, as the number of MDs increases, eventually, the number of tasks executed locally will exceed the number of tasks offloaded to PVs.

Figure 6 shows the total workload allocated to the MEC server under the three schemes. In scheme 1, no computation tasks are offloaded, so the burden on the MEC server is 0. The workload allocated to the MEC server in scheme 3 is lower than that in scheme 2, because PVs share some of the computation tasks. When the number of MDs is 30, scheme 3 reduces the workload for MS by 5%.

Then, we fixed the number of MDs to 30 to observe the change in the metrics as the number of PVs increases.

In Figure 7, the average delay, energy consumption, and overhead of the three schemes are compared with different numbers of PVs. Scheme 3 outperforms both schemes 1 and 2. This is because as the density of PVs increases, it leads to higher utilization of idle computing resources of PVs.

Figure 8 shows the results of task allocation for the proposed scheme. As the number of PVs increases, the number of tasks offloaded to PVs also increases, while the

number of locally executed tasks continues to decrease. From another perspective, as the density of PVs increases, more MDs are likely to be connected to PVs, so that MDs that have given up offloading to MS have more opportunities to offload. In addition, the computational power of the MS is much greater than that of PVs. Tasks are assigned to PVs only when MS cannot serve more tasks. Therefore, the number of tasks executed by MS will not decrease as the number of PVs increases.

7. Conclusion

In this study, we proposed a parked vehicle-assisted mobile edge computing architecture that enhances the task processing capability of MEC servers and improves the resource utilization of parked vehicles. In this work, we first discussed in detail the design principles behind the system model of PV-assisted MEC architecture, which served as a premise for the formulation of computation offloading scheme. Next, by formulating the computation offloading problem as a noncooperative game, we proposed a PV-assisted MEC computation offloading scheme that effectively reduces the burden on the MEC server. Simulation results confirmed the feasibility and high efficiency of the proposed computation offloading scheme. As mentioned in Section 3, incentives are not considered in this study; thus in the future, we will further investigate how to incorporate incentives into the PV-assisted MEC task offloading scheme proposed in this study. Deep reinforcement learning-based techniques have obvious advantages when the problem size is large or when there are multiple conflicting offloading goals [6, 32]. Therefore, another feasible research direction is to apply deep reinforcement learning to further improve the task offloading scheme proposed in this study.

Data Availability

The data used to support the findings of this study are simulated by the algorithm proposed in this article, and the parameters used in the simulation are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Qin Xin Talents Cultivation Program, Beijing Information Science & Technology University (No. QXTCP C202111).

References

- [1] Y. Chen, F. Zhao, X. Chen, and Y. Wu, "Efficient multi-vehicle task offloading for mobile edge computing in 6g networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4584–4595, 2022.
- [2] 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.
- [3] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.
- [4] Y. Chen, F. Zhao, Y. Lu, and X. Chen, "Dynamic task offloading for mobile edge computing with hybrid energy supply," *Tsinghua Science and Technology*, vol. 2021, Article ID 9010050, 9 pages, 2021.
- [5] Ke Zhang, Y. Mao, and S. Leng, "Sabita Maharjan and Yan Zhang. "Optimal delay constrained offloading for vehicular edge computing networks"" in *Proceedings of the 16th Int. Conf. Commun. (ICC)*, pp. 1–6, IEEE, Paris, France, May 2017.
- [6] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," *International Journal of Communication Systems*, vol. 2022, 2022.
- [7] Z. Zhou, J. Feng, Lu Tan, Y. He, and J. Gong, "An air-ground integration approach for mobile edge computing in IoT," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 40–47, 2018, <https://doi.org/10.1109/mcom.2018.1701111>.
- [8] L. Todd, "Parking management strategies," in *Parking Management Best Practices*, pp. 86–225, Routledge, England, UK, 2018.
- [9] N. Liu, M. Liu, W. Lou, G. Chen, and J. Cao, "Pva in vanets: stopped cars are not silent," in *Proceedings of the IEEE INFOCOM*, pp. 431–435, IEEE, Shanghai, China, April 2011.
- [10] C. Ying, X. Hua, and M. Zhuo, "Cost-efficient edge caching for noma-enabled iot services," *China Communications*, vol. 2022, Article ID 8259817, 9 pages, 2022.
- [11] L. Zhu, M. Li, Z. Zhang, and Z. Qin, "Asap: an anonymous smart-parking and payment scheme in vehicular networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 4, pp. 703–715, 2020.
- [12] J. Huang, Z. Tong, and Z. Feng, "Geographical poi recommendation for internet of things: a federated learning approach using matrix factorization," *International Journal of Communication Systems*, vol. 2022, 2022.
- [13] M. Li, Y. Chen, C. Lal, M. Conti, M. Alazab, and D. Hu, "Eunomia: anonymous and secure vehicular digital forensics based on blockchain," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, p. 1, 2021.
- [14] J. Xu, D. Li, W. Gu, and Y. Chen, "Uav-assisted task offloading for iot in smart buildings and environment via deep reinforcement learning," *Building and Environment*, vol. 222, Article ID 109218, 2022.
- [15] J. Huang, B. Lv, Y. Wu, Y. Chen, and X. Shen, "Dynamic admission control and resource allocation for mobile edge computing enabled small cell network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1964–1973, 2022.
- [16] 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.
- [17] N. Cheng, N. Lu, N. Zhang, X. Zhang, X. S. Shen, and J. W. Mark, "Opportunistic wifi offloading in vehicular environment: a game-theory approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1944–1955, 2016.
- [18] 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, oct 2016.

- [19] H. Ding, C. Zhang, Y. Cai, and Y. Fang, "Smart cities on wheels: a newly emerging vehicular cognitive capability harvesting network for data transportation," *IEEE Wireless Communications*, vol. 25, no. 2, pp. 160–169, 2018.
- [20] 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, 2019.
- [21] 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, jun 2017.
- [22] X. Huang, R. Yu, J. Kang, Y. He, and Y. Zhang, "Exploring mobile edge computing for 5g-enabled software defined vehicular networks," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 55–63, dec 2017.
- [23] 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, aug 2018.
- [24] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, Article ID 11158, 2019.
- [25] X. Huang, R. Yu, J. Liu, and L. Shu, "Parked vehicle edge computing: exploiting opportunistic resources for distributed mobile applications," *IEEE Access*, vol. 6, Article ID 66649, 2018.
- [26] Y. Li, Bo Yang, Z. Chen, C. Chen, and X. Guan, "A contract-stackelberg offloading incentive mechanism for vehicular parked-edge computing networks," in *Proceedings of the 89th Veh. Technol. Conf. (VTC)*, pp. 1–5, IEEE, Kuala Lumpur, Malaysia, April 2019.
- [27] D. Han, W. Chen, and Y. Fang, "A dynamic pricing strategy for vehicle assisted mobile edge computing systems," *IEEE Wireless Commun. Lett.*, vol. 8, no. 2, pp. 420–423, 2019.
- [28] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 2, pp. 976–986, 2019.
- [29] J. F. Nash, "Equilibrium points in n-person games," *Proceedings of the National Academy of Sciences*, vol. 36, no. 1, pp. 48–49, 1950.
- [30] D. Monderer and L. S. Shapley, "Potential g," *Games and Economic Behavior*, vol. 14, no. 1, pp. 124–143, 1996.
- [31] S. Chen, J. Hu, Y. Shi et al., "Vehicle-to-everything (v2x) services supported by LTE-based systems and 5g," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 70–76, 2017.
- [32] Y. Chen, Z. Liu, Y. Zhang, Y. Wu, X. Chen, and L. Zhao, "Deep reinforcement learning-based dynamic resource management for mobile edge computing in industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4925–4934, 2021.