applied sciences

MDPI

*Article*

# Vehicular Edge-Computing Framework for Making Use of Parking and Charging Electric Vehicles

**Qi Deng and Feng Zeng ***

School of Computer Science and Engineering, Central South University, Changsha 410083, China
* Correspondence: fengzeng@csu.edu.cn

**Abstract:** In big cities, there are more and more parking lots and charging piles for electric vehicles, and the resources of parking and charging vehicles can be aggregated to provide strong computing power for vehicular edge computing (VEC). In this paper, we propose a VEC framework that uses charging vehicles in parking lots to assist edge servers in processing computational tasks, and an edge crowdsourcing platform (ECP) is designed to manage and integrate the idle computation resources of electric vehicles in parking lots to provide computation services for requesting vehicles. Based on game theory, we first model the interactions among the edge server, the ECP and the requesting vehicles as a Stackelberg game, and theoretically prove the existence of a Nash equilibrium for this Stackelberg game. Then, a genetic algorithm-based game-strategy solving algorithm is proposed to find the optimal strategy for the edge server and ECP. The simulation results demonstrate that the performance of our proposed solution is better than other traditional solutions. Compared with the solution without ECP, our solution can increase the utilities of the edge server and the requesting vehicle by 13.3% and 10.99%, respectively.

**Keywords:** vehicular edge computing; task offloading; game theory; edge crowdsourcing; parking vehicle



**Citation:** Deng, Q.; Zeng, F. Vehicular Edge-Computing Framework for Making Use of Parking and Charging Electric Vehicles. *Appl. Sci.* **2023**, *13*, 4065. https://doi.org/10.3390/app13064065

Academic Editor: Christos Bouras

Received: 31 January 2023
Revised: 10 March 2023
Accepted: 20 March 2023
Published: 22 March 2023



## 1. Introduction

With the rapid development of information technology, advanced vehicular applications such as autonomous driving, intelligent cockpit, virtual reality, and so on, are emerging. These computation-intensive vehicular applications require a large number of computing resources, and low-response latency should be ensured. Due to the limited computing resources in vehicles, the vehicle side may be impossible to provide high-quality computing services for these applications. Moreover, although the cloud center has a large number of computing resources, it is far away from the vehicles and the network transmission of huge data would bring high response latency. Deploying the servers at the edge of the vehicles can shorten the communication distance and also provide powerful computing capability for these vehicular applications. Consequently, vehicular edge computing (VEC) [1] has emerged, with communication base stations and servers deployed on the sides of the road, which form a vehicular edge network to provide low latency and highly reliable computing services for vehicles.

With the support of VEC, the vehicles can offload computation tasks to the edge servers and then obtain efficient computing services. Generally speaking, the quality of service is affected by the number of deployed edge servers. With the vehicular intelligence applications widely used, the limited resources of edge servers will not be able to meet the increasing demand for computation services [2]. Therefore, it becomes a challenge to expand the VEC resources and optimize the allocation of computing resources. To address the problem of insufficient edge resources in VEC, existing research works dealt with this problem from the following two aspects. Firstly, more lightweight edge servers are deployed. For example, Zhang et al. [3] proposed a method to deploy standby servers in

roadside units (RSUs) to alleviate the shortage of VEC resources. However, this approach requires much cost for deploying the servers, and with the continuous increasing of service demand, the shortage of standby resources will occur once again. Meanwhile, it also results in a surplus of resources during off-peak periods. Secondly, it taps into the underutilized resources of nearby vehicles. This solution does not require the deployment of additional edge servers and uses the large number of idle computing resources provided by nearby vehicles to assist the edge servers in computation offloading.

In recent years, we have found that electric vehicles are not only the traditional vehicles, but are also mobile supercomputers loaded with powerful CPUs, large data-storage units, and advanced communication technology. However, these in-vehicle resources will often be idle and not fully utilized. Therefore, aggregating the idle resources in vehicles for VEC would be one of the ways to effectively extend VEC computing resources. In the literature, Li et al. [4] made use of parking vehicles to participate in task processing. It is impractical for traditional fuel vehicles to provide computation resources in the parking state, which is costly and unsustainable. However, nowadays, the commercial operation of electric vehicles has built a large number of special parking lots for electric vehicles in various areas of the cities, and many charging piles are deployed in the parking lots for vehicle charging. The charging piles can obtain the remaining power and related information of the vehicles, and the infrastructure deployed in these parking lots provides great convenience for parking vehicles to participate in VEC services. Compared with VEC servers, these parking vehicles have cheaper computing resources. Therefore, with the parking vehicles' computing resources used in VEC, vehicle users can not only have more adequate edge computing resources, but also obtain relatively cheaper computing services. However, how to effectively integrate the idle resources of parking vehicles for edge computing is a problem to be solved and which will be studied in this paper.

In previous studies, parking vehicles are mainly used in edge computing for content delivery [5], which may face resource limits for all the participants. As vehicular intelligence applications become more and more popular, the limited edge server resources will not be able to meet the growing demand for computing services. Deploying more lightweight edge servers requires significant costs and results in wasted resources during off-peak hours. Moreover, it is impractical, costly, and unsustainable for traditional fuel vehicles to engage in computing tasks while parked. Electric vehicles are well-resourced, inexpensive, and continuously rechargeable, making them the best choice to assist edge servers.

In general, the electric vehicle commercialization services have a charging management system, which manages the charging piles and vehicles, and collects various information about charging vehicles in the parking lots. Therefore, the management system can aggregate a large number of parking vehicles to participate in VEC. In this paper, we propose a framework in which the vehicles assist the VEC servers to complete the offloaded tasks. The framework establishes an edge crowdsourcing platform (ECP) near the parking lots, and the ECP integrates the functions of the charging management system, which enables the ECP to provide charging management services for the vehicles in the parking lots and to act as a crowdsourcing service platform to assign computing tasks to charging vehicles. The main contributions of this paper are as follows.

(1) We propose a VEC framework that uses electric vehicles in parking lots to assist edge servers in processing computational tasks. An edge crowdsourcing platform (ECP) is built to manage and integrate the idle computational resources of electric vehicles in parking lots, so that it can provide computational services for requesting vehicles.
(2) To maximize the utilities of all participants in VEC, based on the Stackelberg game, we analyze the interactions among the requesting vehicles, the edge server, and the ECP, and theoretically prove the existence of a Nash equilibrium for this Stackelberg game. Then, a game strategy-solving algorithm based on a genetic algorithm is proposed to find the optimal strategies for the edge server and the ECP.

The rest of this paper is structured as follows. In Section 2, we discuss and analyze the latest related research works. Section 3 presents the system model and definitions. Section 4

analyzes the interaction between the VEC server and the ECP based on Stackelberg game. Section 5 shows the simulation experimental results. In the last section, a summary and outlook of our work are given.

## 2. Related Work

Vehicular edge computing can provide efficient computing services for vehicular applications. However, the edge resources are limited, and we should consider the optimal utilities of the computing resources, which has been a hot research topic and extensively studied by many scholars.

Considering the interests of both the service requester and provider, Du et al. [6] presented a dual-end optimization problem that tried to balance the interests of the user side and the server side. On the user side, it is dedicated to optimizing the offloading strategy and the frequency of the local central processor. On the server side, the main focus is on optimizing resource scheduling and service provisioning. Considering vehicular edge computing assisted by solar RSUs, Ku et al. [7] proposed a heuristic algorithm based on dynamic programming, which can jointly perform task partitioning and offloading in real-time, as well as accomplish adaptive decisions at the system and application levels, thus reducing end-to-end latency. However, in the above research works, it is supposed that the edge computing resources are sufficient, which is not the truth in actual situations. Nazar et al. [8] proposed a machine learning-based, region-/context-aware equipped content pre-caching strategy that addresses the problem of pre-caching of early decisions and predicted results.

Some scholars have used optimization methods to find the optimal strategy for computation offloading. Dai et al. [9] proposed a particle swarm optimization-based approach to minimize the energy and resource consumption in computation offloading with the delay constraint. Li et al. [10] considered the long-term scheduling policy optimization, and modeled the problem as a specific Markov decision process (MDP) based on the scheduling queue. Then, a deep reinforcement learning (DRL)-based algorithm was proposed to find the solution. Liu et al. [11] developed an efficient task = scheduling algorithm, and the basic idea is to prioritize multiple applications and tasks to ensure the completion time constraints of the applications and the processing dependency requirements of the tasks. Luo et al. [12] further modeled the resource scheduling as a deep reinforcement learning problem that is solved by an enhanced deep Q-network (DQN) [13] algorithm with an independent target Q-network to achieve optimal resource utilization for the whole system.

Some scholars have considered the balance of energy consumption and latency in computation offloading. Ning et al. [14] proposed an online learning model for latency and energy consumption to minimize the offloading cost. Jang et al. [15] considered the changes in the communication environment due to vehicle movement and jointly optimized multiple vehicle offloading and bit scheduling to reduce the total vehicle energy overhead. Zhou et al. [16] proposed a value-based iterative reinforcement learning (RL) approach to determine a joint strategy for computation offloading and resource allocation, a double deep Q network (DDQN) based approach was developed to minimize the energy consumption of the whole system. Kumar et al. [17] summarized the methods related to reducing the energy consumption in wireless sensor networks, and lower energy consumption in sensor networks can save the computational cost in the VEC framework to a great extent, which makes it possible to apply the framework in reality.

In order to improve the performance of VEC, some scholars have designed incentive mechanisms to encourage vehicles to participate in edge computing and contribute their idle resources. Shi et al. [18] proposed a distributed vehicle-vehicle offloading architecture that maps task scheduling into a sequential decision problem considering the channel state and the limited resources. In addition, a dynamic pricing scheme was proposed to guide vehicles to contribute the idle computing resources. Considering the importance of service caching in VEC, Zeng et al. [19,20] proposed a vehicular edge computing framework based on software defined networks, which introduced the reputation to measure the contribution

of each vehicle for VEC. Li et al. [21] designed a VEC architecture using idle resources from parked vehicles. They proposed a three-stage offloading incentive mechanism based on the Starkelberg model and obtained the optimal policy for each participant by solving the three-stage optimal policy using backward induction. The above research work does not mention the organization and management of parked vehicles, and some actual factors are not taken into account. For comparison purposes, we summarize the characteristics of the aforementioned works in Table 1.

**Table 1.** Summary of related works.

| Literature | Problems and Solutions | Results |
| --- | --- | --- |
| Du [6] | A dual-end optimization method is proposed to balance the benefits of the user side and server side. | Reduce costs. |
| Ku [7] | A heuristic algorithm based on dynamic programming is proposed, which can jointly perform task partitioning and offloading in real time, as well as accomplish adaptive decisions at the system and application levels. | Reduce end-to-end latency. |
| Nazar [8] | A machine learning-based, region-/context-aware equipped content pre-caching strategy is proposed that addresses the problem of pre-caching of early decisions and predicted results. | Improved pre-caching in VANET to avoid network congestion. |
| Dai [9] | A particle swarm optimization-based approach is proposed to minimize the energy and resource consumption of computational offloading under delay constraints. | Reduce average task completion time and improve resource utilization. |
| Li [10] | A Deep Reinforcement Learning (DRL) based algorithm is proposed to solve the scheduling policy optimization problem. | Enable tasks with varying degrees of urgency to be completed within time constraints. |
| Liu [11] | A task scheduling algorithm is proposed to rank tasks in order of priority to meet the time constraints and processing dependency requirements of the tasks. | Reduce the average completion time of multiple tasks. |
| Luo [12] | The problem of resource scheduling is addressed by the Deep Q-Network (DQN) algorithm enhanced in deep reinforcement learning. | Achieve optimal utilization of resources. |
| Ning [14] | Limited cellular spectrum and energy supplies restricted. | Minimize the offloading cost. |
| Jang [15] | Address the issue of optimal energy-efficient offloading strategies when the communication environment changes due to vehicles movement. | Reduce total vehicles energy consumption. |
| Zhou [16] | A value-based iterative reinforcement learning (RL) approach, named Q-learning, is proposed to determine a joint strategy for computational offloading and resource allocation. | Minimize the energy consumption of the whole system. |
| Kumar [17] | Methods related to the reduction of energy consumption in WSNs (a group of sensor nodes capable of sensing various environmental parameters) are summarized. | If applied to the VEC framework, resource consumption can be reduced. |
| Shi [18] | The sequential decision problem of task assignment is solved by reinforcement learning; a dynamic pricing scheme is proposed to guide vehicles to contribute idle computational resources. | Improve task assignment performance. |
| Zeng [19] | A software defined network based vehicular edge computing framework is proposed that introduces reputation values to measure the contribution of each vehicle to the VEC. | Reduce delay, increase edge server profits. |
| Li [21] | A three-stage offloading incentive mechanism based on the Starkelberg model is proposed to obtain the optimal strategy for each participant. | Enhance the utility of tripartite participants. |

Different from the existing works, in this paper, we consider the actual scenario of commercialization of new emerging electric vehicle services and try to maximize the utilization of the idle resources of parked vehicles. With the rapid development and popularity of

new energy or electric vehicles, more and more parking and charging sites have been built in cities, especially in the big cities in China. According to the urban development plan of Beijing, there will be 700,000 electric vehicle charging piles in the city by 2025, and the average service radius of electric vehicle public charging facilities in plain areas will be less than 3 km. In the parking and charging lots, the charging piles can collect the information of remaining electric power in each charging vehicle, and control the charging power for the vehicle. Meanwhile, the service providers have developed a management software system to manage and schedule the vehicles for optimal charging. Due to the sufficient parking and charging vehicles in a city, it becomes possible to aggregate the idle computing resources of these parking electric vehicles using crowdsourcing technology, and these parking vehicles can make full use of the idle resources to provide computing services for vehicular applications. Therefore, in this paper, we will establish an edge crowdsourcing platform (ECP) in the parking lot, and the ECP integrates the charging management and crowdsourcing functions. With the ECP, we can design an effective incentive mechanism to encourage parking vehicles to participate in VEC. As a result, the VEC services can be provided via collaboration among the edge servers, the ECP, and the parking vehicles. However, considering the selfishness of service providers, a reasonable incentive mechanism is needed to maximize their respective interests while providing efficient computing services. In the following, we will analyze the three-party cooperation mechanism based on the Stackelberg game theory and find the optimal strategies for the three parties.

## 3. System Model and Definitions

### 3.1. System Model

Considering the full use of resources in vehicles, we propose a VEC framework based on software defined network (SDN) [22], which is shown in Figure 1. The framework includes requesting vehicles, roadside units (RSUs), VEC servers, parking vehicles, and the edge crowdsourcing platform (ECP).

(1) Requesting vehicle: requesting vehicles are those vehicles that need to offload some of their computing tasks to the edge nodes. The requesting vehicles offload some computing tasks to the edge servers and the ECP via RSUs, while they should pay the service providers for the computation offloading.

(2) VEC server: VEC servers are the edge servers deployed on both sides of the road and communicate with requesting vehicles via RSUs. When the requesting vehicles offload the computation task to the VEC server, the VEC server will allocate the resources to complete the task and return the computation results to the requesting vehicles. Providing computation services for requesting vehicles, the VEC servers can obtain the payment from the requesting vehicles.

(3) Edge crowdsourcing platform (ECP): ECP aims to gather parking electric vehicles and use the idle resources of parking vehicles for VEC. The ECP can provide charging management services for the vehicles in the parking lots, meanwhile act as a crowdsourcing platform to assign computing tasks to parking vehicles. Since the number of parking vehicles is large at any time in a city, the ECP can integrate the efficient computing resources for VEC, which ensures that the ECP can continuously and steadily provide computing services for requesting vehicles. As the service provider, the ECP can obtain the reward from the requesting vehicles. Obviously, the computation task is finally undertaken by the parking vehicles elected by the ECP, and the ECP should pay the task undertaking vehicles the corresponding rewards.

(4) Parking vehicles: in this paper, the parking vehicles are those charging electric vehicles in parking lots. In the actual situation, since the charging vehicles have idle computing resources and they will keep static for some time, those vehicles can provide cost-effective computation services for other vehicles. Parking vehicles can join the ECP voluntarily and can use their idle resources to process the tasks assigned by the ECP during the charging process, and finally receive the corresponding rewards.

(5) Software-defined network (SDN): It is an implementation of network virtualization, separating the control plane of network devices from the data plane, controlling the behavior of SDN devices through the standardized interface protocol of OpenFlow, and making decision analysis according to user requirements. Based on SDN technology, the SDN controller can collect information of the requests from vehicles and the status of edge servers and the ECP, then make the optimal schedule of the global computing resources including the edge servers and the ECP. The proposed algorithm can be run in the SDN controller to find the strategies for the requesting vehicles, edge servers, and the ECP, then they can cooperate to achieve their maximized utilities.

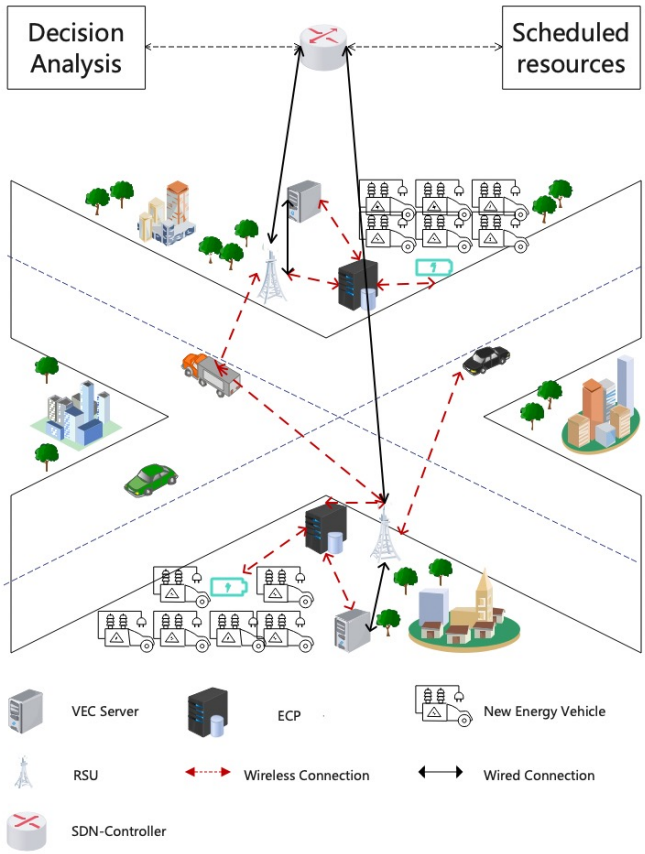


**Figure 1.** The SDN-based VEC framework.

As is shown in Figure 1, some RSUs are deployed on both sides of the road, and each RSU is integrated with an edge server, and all servers, the ECP and SDN controller, are connected in a backbone network. The vehicles can access the edge servers and the ECP via RSUs or charging piles. It is supposed that there are $n$ requesting vehicles within the communication range of RSUs, and the requesting vehicles are denoted as $R_i(i \in [1, n])$. A computation task from the requesting vehicle can be denoted as $A_i = \{Q_i, q_{iv}, q_{ie}\}(i \in [1, n])$, where $Q_i$ denotes the total task size which may be the data volume to be processed, the $q_{iv}$ denotes the task size offloaded to the VEC server, and $q_{ie}$ denotes the task size offloaded to the ECP. If the ECP is allocated a task, the ECP will post the task on the platform to recruit $m$ parking vehicles for crowdsourcing, and each one requires a fee denoted as $P_j(j \in [1, m])$. Then, the $m$ parking vehicles will collaborate to process the task. We assume that each parking vehicle is assigned a workload denoted as $q_{ij}$.

As shown in Figure 2, the computation offloading includes the following steps:

*Step 1:* The vehicles request the edge network for computation task offloading.

*Step 2:* The SDN controller executes the algorithm to determine the best strategies for the edge server and the ECP, and responds to the requesting vehicle.

*Step 3:* According to the responded solution, the requesting vehicle offloads the subtasks to the edge server and the ECP via RSU.

*Step 4:* The edge server processes the offloaded subtask and receives the reward. With the crowdsourcing platform, the ECP assigns the sub-tasks to the selected parking vehicles in an efficient manner, and the ECP pays the corresponding rewards to the task-undertaking vehicles after receiving the payoffs from the requesting vehicle.

*Step 5:* The SDN controller finds the location of the requesting vehicle and sends back the computation results to the requesting vehicle via its closest RSU.

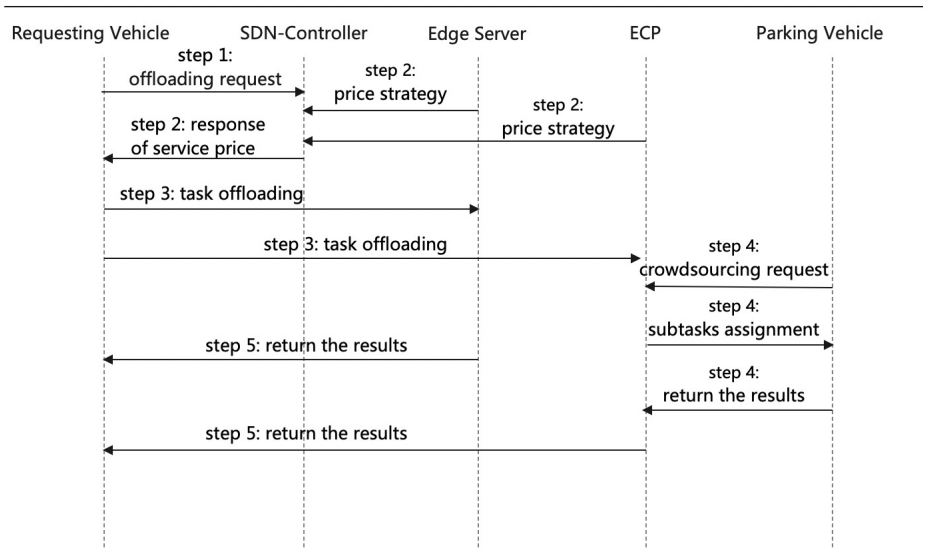


**Figure 2.** Task offloading steps in parking vehicles assisted edge computing.

### *3.2. Delay and Energy Consumption Model*

The total delay of task offloading can be divided into two parts, which are task data transmission time and task data processing time. The time of task data transfer from the requesting vehicles ($R_i$) to the edge server is denoted as $T_{iv}^t = \frac{q_{iv}}{r_v}$, and the time of task data transmission to the ECP is denoted as $T_{ie}^t = \frac{q_{ie}}{r_e}$, where $r_v$ and $r_e$ denotes the data transmission rate from the requesting vehicle to the edge server and the ECP, respectively. The time of the ECP to assign the computation subtasks to the parking vehicles ($P_j$) is denoted as $T_{ij}^t = \frac{q_{ie}}{d}$ , where $d$ is the data transmission rate between the ECP and the parking vehicle.

For any computation task, the processing time is related to the volume of task data. Then, the task processing time of the edge server can be expressed as $T_{iv}^c = \frac{\alpha q_{iv}}{c_v}$, where $\alpha$ is the number of CPU cycles for unit data processing, and $C_v$ is the data volume processed by the edge server in every unit time. Correspondingly, the task processing time of the parking vehicles in the ECP can be expressed as $T_{ij}^c = \frac{\alpha q_{ie}}{c_{je}}$, where $C_{je}$ denotes the computing capacity of the $j$ parking vehicle. Since the data volume of the result is small, the return time of the computation result is ignored in this paper. Therefore, the delay for the requesting vehicle offloading task to the edge server and the ECP can be expressed as $T_{iv}$ and $T_{ij}$:

$$T_{iv} = T_{iv}^t + T_{iv}^c = \frac{q_{iv}}{r_v} + \frac{\alpha q_{iv}}{c_v}, \tag{1}$$

$$T_{ij} = T_{ie}^t + T_{ij}^c + T_{ij}^t = \frac{q_{ie}}{r_e} + \frac{\alpha q_{ie}}{c_{je}} + \frac{q_{ie}}{d}. \tag{2}$$

As far as the energy consumption is concerned, the energy consumption of ECP is mainly for assigning the computation subtasks to the parking vehicles, which can be defined as:

$$E_{ie}^{d} = \sigma \frac{q_{ie}}{d}, \tag{3}$$

where $\sigma$ denotes the task assignment energy factor of the ECP.

The energy consumption for data transmission includes the transmission consumption from the requesting vehicle to the edge server and the ECP, respectively, which are defined as $E_{iv}^{t}$ and $E_{ie}^{t}$:

$$E_{iv}^{t} = \rho T_{iv}^{t} = \rho \frac{q_{iv}}{r_v}, \tag{4}$$

$$E_{ie}^{t} = \rho T_{ie}^{t} = \rho \frac{q_{ie}}{r_e}, \tag{5}$$

where $\rho$ is the energy consumption coefficient of data transmission for the requesting vehicles.

After receiving the task data, the edge server and the ECP perform task execution and return the results to the requesting vehicles. Since the energy consumption of data reception is much smaller than the energy consumption of data transmission, the data volume of the result is small. Therefore, in this paper, we do not consider the energy consumption of data reception and the energy consumption of returning the results. Consequently, the energy consumption of the edge servers and the parking vehicles can be expressed as

$$E_{iv}^{c} = \phi T_{iv}^{c} = \phi \frac{\alpha q_{iv}}{c_v}, \tag{6}$$

$$E_{ij}^{c} = \phi T_{ij}^{c} = \phi \frac{\alpha q_{ij}}{C_{je}}, \tag{7}$$

where $\phi$ denotes the energy consumption factor for task processing.

### 3.3. Utility Function for Requesting Vehicle

Generally speaking, with the payment for the computation offloading service, the requesting vehicle expects to obtain high service satisfaction. Consequently, the utility function of the requesting vehicle can be defined as the service satisfaction minus the payment and its data transmission energy consumption. We assume that $p_{iv}$ and $p_{ie}$ denote the service prices given by the edge server and the ECP, respectively. Then, the utility function of the requesting vehicle can be defined as $U_{R_i}$:

$$U_{R_i} = D_{R_i} - p_{iv} q_{iv} - p_{ie} q_{ie} - \left(E_{iv}^{t} + E_{ie}^{t}\right), \tag{8}$$

where $D_{\text{req}}$ is the satisfaction function of the requesting vehicle, which is determined by the allocated computation resources and the task processing delay obtained by the requesting vehicle, denoted as:

$$\begin{aligned} D_{R_i} &= \omega \left[ (1 - \mu T_{\text{iv}}^{c}) q_{iv} + \left(1 - \mu T_{ij}^{c}\right) q_{ie} \right] \\ &= \omega \left[ \left(1 - \frac{\mu \alpha q_{iv}}{C_v}\right) q_{iv} + \left(1 - \frac{\mu \alpha (Q_i - q_{iv})}{C_{je}}\right) (Q_i - q_{iv}) \right] \\ &= \omega \left[ Q_i - \frac{\mu \alpha Q_i^2}{C_{je}} + \frac{2 \mu \alpha Q_i q_{iv}}{C_{je}} - \left(\frac{\mu \alpha}{C_v} + \frac{\mu \alpha}{C_{je}}\right) {q_{iv}}^2 \right], \end{aligned} \tag{9}$$

where $\mu$ is the delay factor of the edge nodes. The shorter the delay, the higher the satisfaction. The $\omega$ denotes the satisfaction coefficient of the requesting vehicles. Therefore, the utility function of the requesting vehicle is expressed as:

$$U_{R_i} = \omega \left[ Q_i - \frac{\mu \alpha Q_i^2}{C_{je}} + \frac{2 \mu \alpha Q_i q_{iv}}{C_{je}} - \left(\frac{\mu \alpha}{C_v} + \frac{\mu \alpha}{C_{je}}\right) q_{iv}^2 \right] - p_{iv} q_{iv} - p_{ie} (Q_i - q_{iv}) - \rho \left( \frac{q_{iv}}{r_v} + \frac{Q_i - q_{iv}}{r_e} \right). \tag{10}$$

### 3.4. Utility Function for Edge Server

As the service provider, the edge servers receive the payment from the requesting vehicle at the cost of computing energy consumption. Consequently, the utility function of an edge server can be defined as the received reward from providing computing services minus its computing energy consumption, which is expressed as:

$$U_{VEC} = \sum_{i=1}^{n}(p_{iv}q_{iv} - E_{iv}^{c}) = \sum_{i=1}^{n}\left(p_{iv}q_{iv} - \phi\frac{\alpha q_{iv}}{C_v}\right). \tag{11}$$

### 3.5. Utility Function for ECP

As the crowdsourcing platform, the ECP receives the offloaded task from the requesting vehicle, and the offloaded task can be divided into some subtasks which will be distributed to some parking vehicles for processing. When the task is completed and the results are returned to the requesting vehicle, the ECP will receive the reward from the requesting vehicle, and part of the reward should be transferred to the participating parking vehicles. Consequently, the utility function of the ECP is defined as the received reward minus the reward transferred to the parking vehicles, which can be expressed as:

$$U_{ECP} = \sum_{i=1}^{n}\left(p_{ie}q_{ie} - \sigma\frac{q_{ie}}{d} - p_{new}q_{ie}\right), \tag{12}$$

where $p_{new}$ represents the price of task processing for parking vehicles.

### 3.6. Price Function for Parking Vehicle

With the support of the crowdsourcing platform, the parking vehicles will cooperate to complete the task offloaded to the ECP. The utility function of the parking vehicle is defined as the received reward minus its energy consumption for task processing, which is denoted as:

$$U_{P_j} = \sum_{i=1}^{n}\left(p_{new}q_{ij} - \phi\frac{\alpha q_{ij}}{c_{je}}\right). \tag{13}$$

The meanings of the above parameters are shown in Table 2.

**Table 2.** Main notation used in this paper.

| Parameters | Meaning |
|---|---|
| $Q_i$ | Total task quantity |
| $q_{iv}$ | Amount of tasks offloaded to VEC severs |
| $q_{ie}$ | Amount of tasks offloaded to ECP |
| $q_{ij}$ | Amount of tasks offloaded to parking vehicle $P_j$ |
| $r_v$ | Rate of task transmission from the requesting vehicles to the VEC servers |
| $r_e$ | Rate of task transmission from the requesting vehicles to the ECP |
| $d$ | Task transfer rate of ECP |
| $C_v$ | Computing capability of VEC servers |
| $C_{je}$ | Computing capability of parking vehicles |
| $p_{iv}$ | Price per unit resource for VEC servers |
| $p_{ie}$ | Price per unit resource for ECP |
| $p_{new}$ | Price per unit resource for parking vehicles |
| $\alpha$ | Number of CPU cycles required to process a one-byte task |
| $\sigma$ | Energy consumption factor for task assignment of ECP |
| $\rho$ | Transmission energy factor for requesting vehicles |
| $\varphi$ | Energy consumption factor of edge nodes |
| $\mu$ | Time processing factor of edge nodes |
| $\omega$ | Satisfaction factor of requesting vehicle |

## 4. Problem Description and Optimal Solution

### 4.1. Problem Description

There are four types of participants in VEC including the requesting vehicle, the edge server, the ECP, and the parking vehicles, and they all expect to maximize their interests. The interest of a participant has the impact on the others' interests, and there is a competition–cooperation relationship among them, which can be abstracted as the following three problems.

**Problem 1.** *Maximizing the Utility of Requesting Vehicles*

$$\begin{aligned} &\max_{q_{iv}, p_{iv}, p_{ie}} U_{R_i}(q_{iv}^*, p_{iv}^*, p_{ie}^*) \\ &s.t.\ Q_i - q_{iv}^* \geq 0 \end{aligned} \tag{14}$$

In (14), the $q_{iv}^*$ denotes the optimal amount of tasks offloaded to the edge server, and the $p_{iv}^*$ and $p_{ie}^*$ are the optimal prices of the edge server and ECP, respectively. The constraint indicates that the optimal task volume offload to the edge server cannot be greater than the total volume of the offloaded tasks.

**Problem 2.** *Maximizing the Utility of the Edge Servers*

$$\begin{aligned} &\max_{q_{iv}, p_{iv}} U_{VEC}(q_{iv}^*, p_{iv}^*) \\ &s.t.\ p_{iv} > \frac{\varphi\alpha}{c_v} \end{aligned} \tag{15}$$

In (15), the constraint indicates that the reward received by the edge server should be greater than the cost of task processing.

**Problem 3.** *Maximizing the Utility of ECP*

$$\begin{aligned} &\max_{q_{ie}, p_{ie}} U_{ECP}(q_{ie}^*, p_{ie}^*) \\ &s.\ t.\ C_1 : q_{ie}^* = Q_i - q_{iv}^* \\ &\quad C_2 : p_{ie}^* > \frac{\sigma}{d} + p_{new} \end{aligned} \tag{16}$$

In (16), the first constraint indicates that the sum of the tasks that the requesting vehicles offloaded to the ECP and the edge servers are equal to the total task volume, and the second constraint indicates that the reward received by the ECP is greater than the sum of its energy consumption and the payment to the parking vehicles.

### 4.2. Task Offloading Analysis Based on Starkelberg Game

As mentioned above, there is a competition–cooperation relationship between the participants in VEC, and each participant wants to maximize its interest. We can model the interactions among the participants as a Starkelberg game, which is a two-stage full information dynamic game with sequential timing. The participants in the game are divided into two groups, the leaders and the followers. The leaders are the participants who make decisions first, and the remaining participants make decisions based on the leaders' decisions which are called followers. The main idea of the Starkelberg game is that each side chooses its strategy based on the possible strategies of the other side to ensure that it maximizes the benefit of the other side's strategy, thus achieving a Nash equilibrium [23].

As shown in Figure 2, the framework proposed in this paper is a three-party game model based on the Starkelberg game. The participants are the requesting vehicle, the edge server, and the ECP. The edge server and the ECP, as the leaders of the game, make the price decision first, and the requesting vehicle, as the follower, makes the offloading

decision later. Since the edge server has higher processing power than the parking vehicle, the requesting vehicle will offload the task to the edge server first. At the same time, the requesting vehicle can also offload tasks to the ECP to get relatively cheaper services. Once the ECP receives the task offload request, the ECP will divide the task into some sub-tasks and assign these sub-tasks to the parking vehicle for processing. In addition, there is a non-cooperative game between the edge server and the ECP. For example, when the VEC server offers a higher price, the requesting vehicle is more likely to offload more tasks to the ECP, even if the ECP has a lower computation rate. Likewise, when the ECP offers a higher price, the requesting vehicle is more likely to obtain compute services from the VEC server. Thus, the pricing strategies of the edge server and the ECP can influence each other. In this three-party game, each party is a rational, selfish individual who wants to achieve maximum utility. From the above, we can know that this game has obvious leaders and followers, and it is suitable to use Starkelberg game to analyze their behavior interaction. In the following, based on the inverse induction method, we will prove that there is a unique Nash equilibrium of the game, and find the optimal strategies for the participants in the game.

**Definition 1.** *It is supposed that the best strategy for requesting vehicles is* $(q_{iv}^*, q_{ie}^*)$ *which means the* $q_{iv}^*$ *and* $q_{ie}^*$ *task data amount are offloaded to the edge server and the ECP, respectively. Moreover,* $p_{iv}^*$ *and* $p_{ie}^*$ *are the best prices given by the edge server and the ECP, respectively. Then, for any combination of* $p_{iv}$, $p_{ie}$, $q_{iv}$ *and* $q_{ie}$, *if the inequality* (17) *holds, the* $(p_{iv}^*, p_{ie}^*, q_{iv}^*)$ *is the solution of Stankelberg game equilibrium.*

$$\begin{cases} U_{R_i}(q_{iv}^*, p_{iv}^*, p_{ie}^*) \geq U_{R_i}(q_{iv}, p_{iv}, p_{ie}) \\ U_{VEC}(q_{iv}^*, p_{iv}^*) \geq U_{VEC}(q_{iv}, p_{iv}) \\ U_{ECP}(q_{ie}^*, p_{ie}^*) \geq U_{ECP}(q_{ie}, p_{ie}) \end{cases} \tag{17}$$

*In this two-stage game, we use the inverse induction method to analyze the interactions between participants, and the second stage is analyzed first.*

4.2.1. Second Stage Starkelberg Game

We first analyze the utility of the requesting vehicle, and we have the first and second derivatives of (10):

$$\frac{\partial D_{R_i}}{\partial q_{iv}} = \omega \left[ \frac{2\mu\alpha Q_i}{C_{je}} - 2q_{iv}\left( \frac{\mu\alpha}{C_v} + \frac{\mu\alpha}{C_{je}} \right) \right] - p_{iv} + p_{ie} - \gamma \left( \frac{\rho}{r_v} - \frac{\rho}{r_e} \right), \tag{18}$$

$$\frac{\partial^2 D_{R_i}}{\partial q_{iv}} = -2\omega \left( \frac{\mu\alpha}{C_v} + \frac{\mu\alpha}{c_{je}} \right) < 0. \tag{19}$$

As is shown in (19), the second derivative of the utility function of the requesting vehicle is constantly less than zero, thus the utility function of the requesting vehicle is strictly concave, and the function has a maximum value. Therefore, it can be proved that the requesting vehicle can provide the best task offloading strategy regardless of the price strategies given by the edge server and the ECP.

Let the first derivative of the utility function of the requesting vehicles be equal to zero, so we set (18) to be zero, then we can find the optimal offloading strategy of the requesting vehicle. Therefore, we can obtain:

$$q_{iv}^* = \frac{Q_i c_v}{c_{je} + c_v} - \frac{c_v c_{je}[r_v r_e(p_{iv} - p_{ie}) + \rho(r_v - r_e)]}{2\omega\mu\alpha(c_{je} + c_v)}. \tag{20}$$

If the optimal amount of tasks offloaded to the edge server is less than the total amount of tasks, the rest amount of the tasks should be offloaded to the ECP, so that we have:

$$\begin{cases} q_{iv}^* = \dfrac{Q_i c_v}{c_{je}+c_v} - \dfrac{c_v c_{je}[r_v r_e(p_{iv}-p_{ie})+\rho(r_v-r_e)]}{2\omega\mu\alpha(c_{je}+c_v)} \\ q_{ie}^* = Q_i - q_{iv}^* \\ Q_i > q_{iv}^* \end{cases}. \tag{21}$$

Otherwise, the optimal offloading strategy for the requesting vehicle is expressed as:

$$\begin{cases} q_{iv}^* = Q_i \\ q_{ie}^* = 0 \\ Q_i \le q_{iv}^* \end{cases}. \tag{22}$$

4.2.2. First Stage Starkelberg Game

With the offloading strategy given by the requesting vehicle, the utility function of the edge server (Equation (11)) can be changed into (23):

$$U_{VEC} = \sum_{i=1}^{n}\left[\left(\frac{Q_i C_v}{C_{je}+C_v} - \frac{C_v C_{je}[r_v r_e(p_{iv}-p_{ie})]}{2\omega\mu\alpha(C_{je}+C_v)} - \frac{\rho(r_v-r_e)}{2\omega\mu\alpha(C_{je}+C_v)}\right)\left(p_{iv}-\frac{\varphi\alpha}{C_v}\right)\right]. \tag{23}$$

Then, we can have the first and second derivatives of (23), expressed as (24) and (25), respectively.

$$\frac{\partial U_{VEC}}{\partial p_{iv}} = \sum_{i=1}^{n}\left[\left(\frac{Q_i C_v}{C_{je}+C_v} - \frac{C_v C_{je}[r_v r_e(p_{iv}-p_{ie})]}{2\omega\mu\alpha(C_{je}+C_v)} - \frac{\rho(r_v-r_e)}{2\omega\mu\alpha(C_{je}+C_v)}\right) - \frac{C_v C_{je} r_v r_e}{2\omega\mu\alpha(C_{je}+C_v)}\left(p_{iv}-\frac{\varphi\alpha}{C_v}\right)\right]. \tag{24}$$

$$\frac{\partial^2 U_{VEC}}{\partial {p_{iv}}^2} = -\frac{C_v c_{je} r_v r_e}{\omega\mu\alpha(C_{je}+C_v)} < 0. \tag{25}$$

The above (25) illustrates that the second derivative of the utility function of the VEC server is constantly less than zero, which means that the utility function is strictly concave and has a maximum value. Therefore, the edge server can have the optimal price strategy. Let (24) to be zero, and we have the optimal price strategy of the edge server as (26):

$$p_{iv}^* = \sum_{i=1}^{n}\left(\frac{2\alpha\mu\omega Q_i c_v + \varphi\alpha r_v r_e C_{je} - \rho(r_v-r_e)C_{je}C_v}{2C_v c_{je} r_v r_e} + \frac{p_{ie}^*}{2}\right). \tag{26}$$

According to (12), (21) and (22), if $Q_i \le q_{iv}^*$, the utility of ECP is zero. Otherwise, the utility function of ECP is expressed as (27):

$$\begin{cases} U_{ECP} = \sum_{i=1}^{n}\left[\left(Q_i - \left(\frac{Q_i C_v}{C_{je}+C_v} - \frac{C_v C_{je} r_v r_e(p_{iv}-p_{ie})}{2\omega\mu\alpha(C_{je}+C_v)} - \frac{\rho(r_v-r_e)}{2\omega\mu\alpha(C_{je}+C_v)}\right)\right)\left(p_{ie}-\frac{\alpha(\varphi+1)}{C_{je}}-\frac{\sigma}{d}\right)\right] \\ Q_i > q_{iv}^* \end{cases}. \tag{27}$$

Then, we have the first and second derivatives of (27), expressed as (28) and (29), respectively.

$$\frac{\partial U_{ECP}}{\partial p_{ie}} = \sum_{i=1}^{n}\left[Q_i - \frac{Q_i C_v}{C_{je}+C_v} + \frac{C_v C_{je} r_v r_e(p_{iv}-p_{ie})}{2\omega\mu\alpha(C_{je}+C_v)} + \frac{\rho(r_v-r_e)}{2\omega\mu\alpha(C_{je}+C_v)} - \frac{C_v C_{je} r_v r_e}{2\omega\mu\alpha(C_{je}+C_v)}\left(p_{ie}-\frac{\alpha(\varphi+1)}{C_{je}}-\frac{\sigma}{d}\right)\right], \tag{28}$$

$$\frac{\partial^2 U_{ECP}}{\partial {p_{ie}}^2} = -\frac{C_v C_{je} r_v r_e}{\omega\mu\alpha(C_{je}+C_v)} < 0. \tag{29}$$

The second derivative (29) is constantly less than zero, which means that the utility function is strictly concave, and the function has a maximum value. Let its first derivative (28) to be zero, we have the optimal price strategy for ECP as (30):

$$\begin{cases} p_{ie}^{*} = \sum_{i=1}^{n} \left( \frac{\omega \mu \alpha Q_i}{C_v r_v r_e} + \frac{\rho(r_v - r_e)}{2 r_v r_e} + \frac{\alpha(\varphi + 1)}{2 C_{je}} + \frac{\sigma}{2d} + \frac{p_{iv}^{*}}{2} \right) \\ Q_i > q_{iv}^{*} \end{cases} . \tag{30}$$

From the above analysis, it is proved that there are optimal strategies for the edge server, the ECP, and the requesting vehicle, and there is a Nash equilibrium in this Stackelberg game.

In order to find the optimal solution, a game strategy algorithm (GSA) based on a genetic algorithm is proposed to solve the Stackelberg equilibrium problem, and the optimal strategies can be obtained by iterative crossover, mutation, selection, and reproduction in the genetic algorithm. The proposed algorithm is shown in Algorithm 1.

**Algorithm 1** GSA (Game Strategy Algorithm)

**Input:** $A_i^{*} = \{Q_i, q_{iv}^{*}, q_{ie}^{*}\} (i \in [1, n])$, population size $N$, cross probability $cp$, mutation probability $mp$, generation gap $gap$, maximum genetic generation $Maxgen$

**Output:** $p_{iv}^{*}$, $p_{ie}^{*}$ $U_{VEC}, U_{ECP}$

```
iteration = 0
U_VEC = Σ_{i=1}^{n} ( p_iv q_iv − φ (α q_iv / C_v) )
U_ECP = Σ_{i=1}^{n} ( p_ie q_ie − σ (q_ie / d) − p_new )
for iteration <Maxgen do
    fit = fitness(U_VEC)
    offspring = select(parent, fitness, gap)
    offspring = intersect (offspring, cp)
    offspring = variation (offspring, mp)
    parent = replace (offspring,U_VEC)
    iteration = iteration + 1
end for
for iteration <Maxgen do
    fit = fitness(U_ECP)
    offspring = select (parent, fitness, gap)
    offspring = intersect (offspring, cp)
    offspring = variation (offspring, mp)
    parent = replace (offspring,U_ECP)
    iteration = iteration + 1
end for
return p*_iv, p*_ie, U_VEC, U_ECP
```

**Theorem 1.** *The time complexity of GSA is O(Maxgen* $\cdot n^2$*).*

**Proof of Theorem 1.** As shown in Algorithm 1, we know that both $\frac{\partial^2 U_{VEC}}{\partial {p_{iv}}^2}$ and $\frac{\partial^2 U_{ECP}}{\partial {p_{ie}}^2}$ are less than zero, $U_{VEC}$ is a concave function about $p_{iv}$, $U_{ECP}$ is a concave function about $p_{ie}$, and the optimal solutions exist for both the functions $U_{VEC}$ and $U_{ECP}$. Therefore, we can use the genetic algorithm to quickly get the optimal strategies for the edge server and the ECP. The core of the genetic algorithm is the dual iteration with time complexity not exceeding $n^2$, and its complexity is determined by the fitness function, in addition, we set the iteration threshold $Maxgen$. Consequently, the time complexity of the algorithm is O($Maxgen \cdot n^2$). □

## 5. Simulations

### *5.1. Simulation Parameters*

In this section, we will use Matlab to evaluate the performance of the model. The total task quantity of the requesting vehicle is 5 Mbits, and the rates of task transmission from the requesting vehicle to the VEC server and the ECP are set to 0.8 Mbps and 0.6 Mbps, respectively. The rate of task transmission from the ECP to the parking vehicles is 0.6 Mbps. The computational capability of the VEC server is set to 3, 3.5, 44, 4.5, and 5. The per resource price of the VEC server is set to 0.6, and the per resource price of the ECP is set to 0.2–0.5. The number of CPU cycles required to process one byte of computation task is set to 0.002. The satisfaction factor of the requesting vehicle is 1.9, the transmission energy factor of the requesting vehicle is 0.8, the time processing factor of the edge nodes is 1.2, and both the task allocation energy factor of the ECP and the computation energy factor of the edge nodes are 1. The main parameters used in the simulation are shown in Table 3.

**Table 3.** The simulation parameters.

| Parameter | Value | Parameter | Value | Parameter | Value |
|---|---|---|---|---|---|
| $Q_i$ | 3 Mbits | $p_{iv}$ | 0.6 | $\varphi$ | 1 |
| $r_v$ | 0.8 Mbps | $p_{ie}$ | $(0.2, 0.5)$ | $\rho$ | 0.8 |
| $r_e$ | 0.6 Mbps | $\alpha$ | 0.002 | $\mu$ | 1.2 |
| $C_v$ | [3, 5] MHz | $\sigma$ | 1 | d | 0.6 Mbps |
| $C_{je}$ | [2, 3] MHz | $\omega$ | 1.9 | | |

### *5.2. Simulation Results*

Figure 3 shows the relationship between task offloading strategy and the processing capability and the prices of service. As we can find in Figure 3, with the increasing of the processing capability of the edge servers, the amount of the tasks offloaded to the edge servers increases accordingly. Moreover, the higher the price of ECP, the higher the number of tasks offloaded to the edge servers.

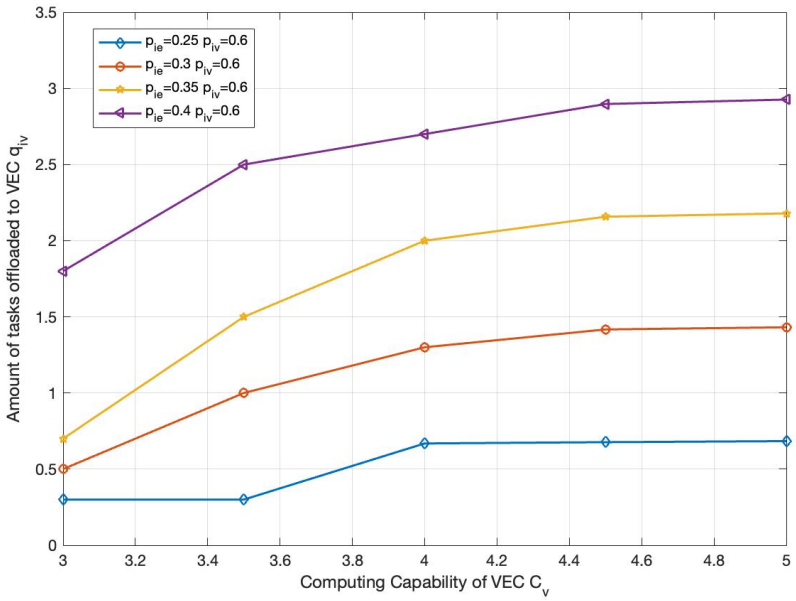


**Figure 3.** Impact of the processing capability and service price on the performance.

Figure 4 demonstrates that, with the incrase of the computing capability of the edge servers, the utility of the requesting vehicle will increase also. When the price of the ECP increases, the utility of the requesting vehicle will decrease. Thus, the simulation results show that the utility of the requesting vehicle increases as the processing capability of the

edge server or ECP rises, and decreases as the service price rises. Moreover, when the processing capability of the edge server increases or the price of the ECP increases, the requesting vehicles are more inclined to obtain resources from the edge servers.

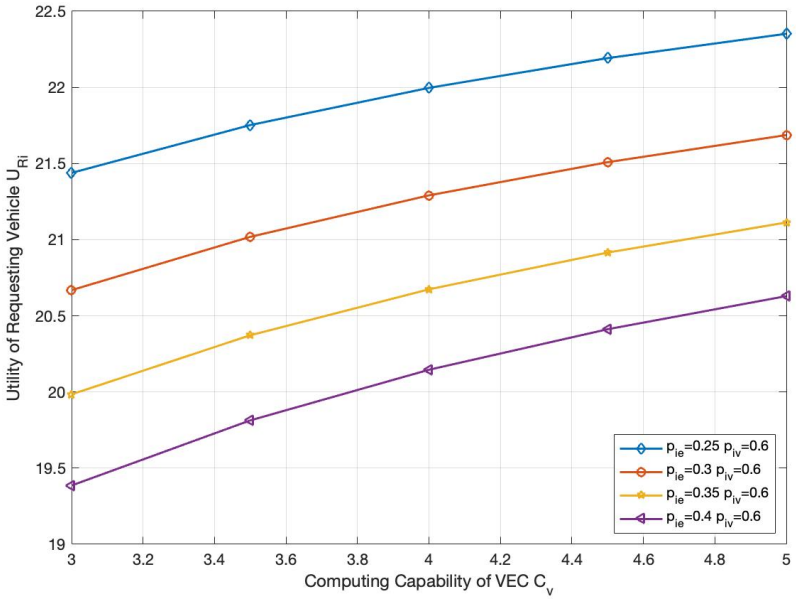


**Figure 4.** Impact of the processing capability and service price on the utility of requesting vehicle.

Then, we compare the proposed model in this paper with two other traditional edge server-based schemes. Our solution is denoted as "VEC+ECP", and the other two are denoted as "VEC" and "VEC+ECP+Random". The "VEC" solution means that the requesting vehicles only request computation service from the edge servers and the parking vehicles do not participate in the task offloading. The "VEC+ECP+Random" solution means that no gaming is performed and the tasks are randomly offloaded to the edge servers and the ECP.

Figure 5 shows the comparison of the utilities of the three schemes for the requesting vehicles when the ECP price changes. We can find that the requesting vehicle has the highest utility in the VEC+ECP solution. When the ECP price is relatively low, both the VEC+ECP and the VEC+ECP+Random have a higher utility of the requesting vehicle than the VEC solution. Conversely, when the price is high, the performance difference between these three solutions becomes smaller and the requesting vehicles may be discouraged from obtaining computation services from the ECP due to the high price.

Figure 6 shows the utility of the requesting vehicle with the increasing processing capability of the edge server in the three solutions. From Figure 6, we can find that our solution has the best performance. Compared with the VEC solution, our solution has the utility of the requesting vehicle increased by 10.99%. When the processing capability of the edge server is relatively high, the requesting vehicles prefer to obtain computation services from the edge servers. For example, when $C_v$ is greater than 3.5, the utility in the VEC solution is higher than that in the VEC+ECP+Random solution. In addition, the utility of the requesting vehicle in this scheme becomes higher as the computational capability increases. This is because the VEC servers and ECPs need to compete with each other and must reduce their prices to attract more requesting vehicles.

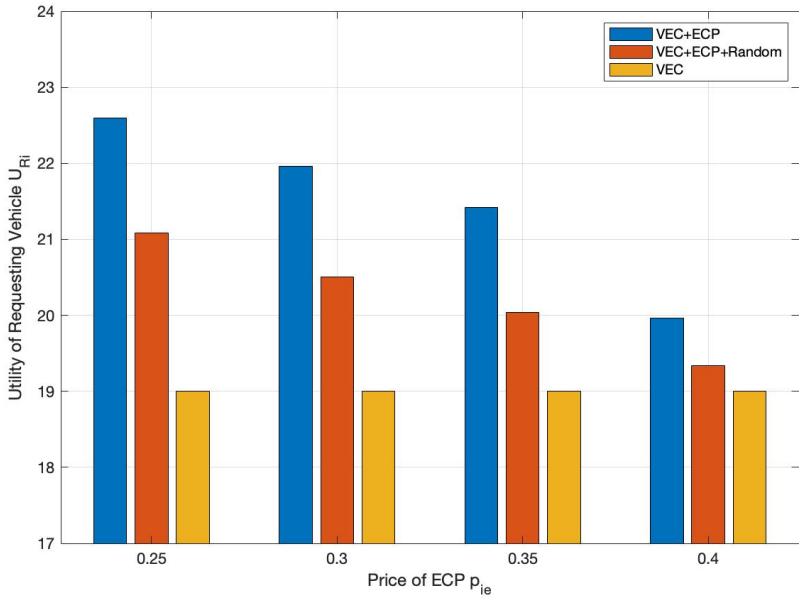


**Figure 5.** Performance comparison for the utility of requesting vehicle.

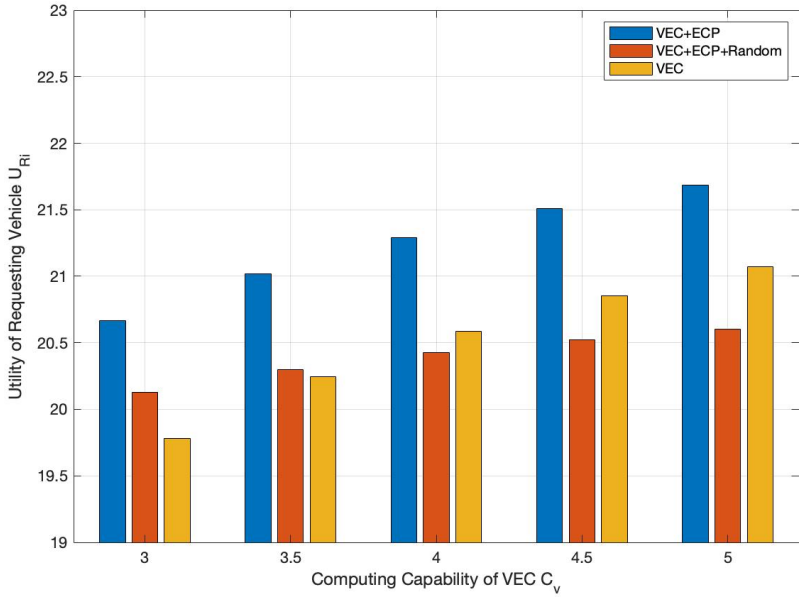


**Figure 6.** Impact of $C_v$ on the utility of requesting vehicle.

Figure 7 shows the process of the GSA algorithm to find the best strategy via iterative optimization. From the figure, it can be seen that the numbers of iterations for the three solutions are 15, 23, and 36, which indicates that the proposed algorithm has better convergence and can find the best strategy quickly. Moreover, the utility of the edge server in the VEC+ECP is higher than that in the VEC+ECP+Random and the VEC, having 13.3% higher utility than that in VEC solution, which means that the edge server will have higher utility with the assistance of parking vehicles.

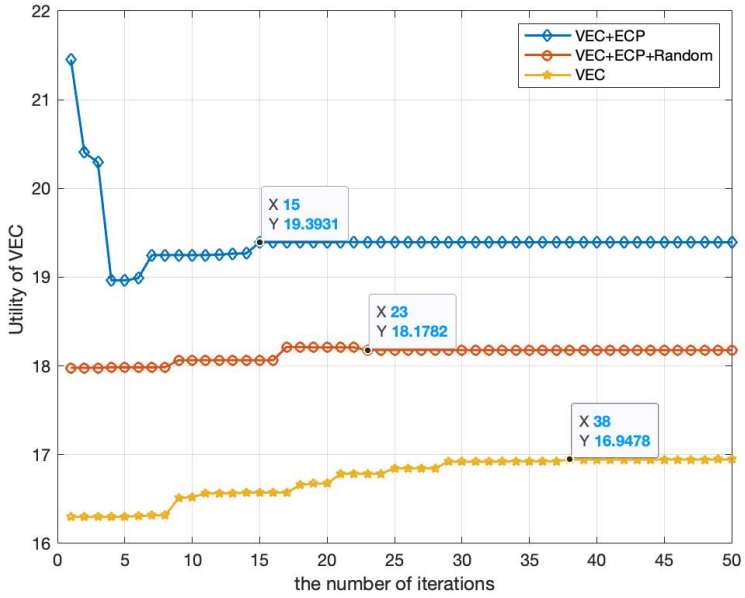


**Figure 7.** GSA algorithm performance.

Figure 8 plots the utilities of the edge server, the ECP, and the requesting vehicle under different $C_v$. In the non-cooperative game, the ECP and the edge server will reduce their prices due to the competition, which eventually leads to lower utilities. Especially, the competition is more intense when the processing capabilities of the edge server and the ECP increases. As a result, the utilities of both the edge server and the ECP decrease. Relative to the VEC servers and ECPs, the utility of the requested vehicles gradually increases because the higher $C_v$ leads to a lower price of the edge nodes.

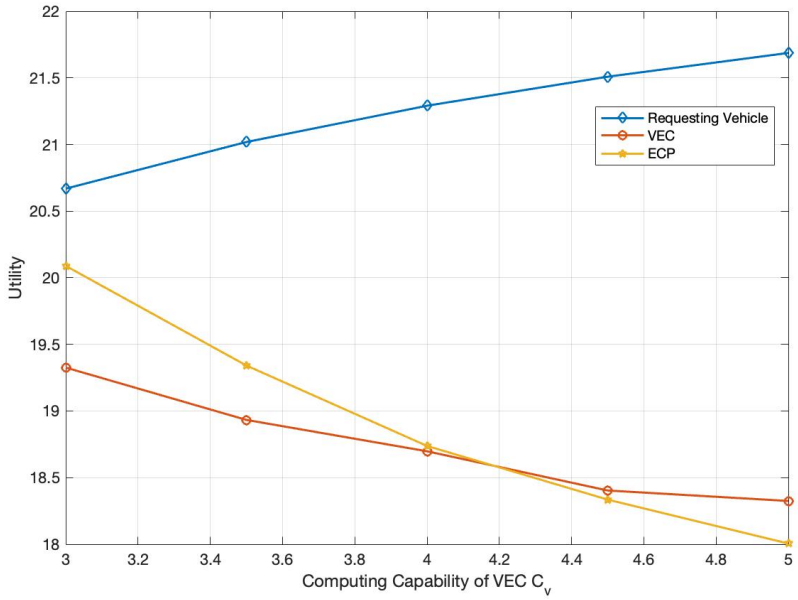


**Figure 8.** The Utilities of the edge server, ECP, and requesting vehicle.

Based on the above simulation results, we can conclude that the performance of the proposed solution is better than other traditional solutions. With the GA-based algorithm and game theory, all three participants can obtain the optimal strategies.

## 6. Conclusions

In this paper, we have proposed a VEC framework that uses electric vehicles in parking lots to assist edge servers in processing computational tasks. We also build an edge crowdsourcing platform (ECP) that aims to manage and integrate the idle computational resources of electric vehicles in parking lots to provide computational services to requesting vehicles in response to the problems of insufficient computational resources of edge servers and the unsuitability of fuel vehicles for practical applications. In addition, based on the Stackelberg game, we have analyzed the interactions among requesting vehicles, the edge server, and the ECP, and theoretically proved the existence of a Nash equilibrium for this Stackelberg game. Then, a game strategy-solving algorithm based on a genetic algorithm is proposed to find the optimal strategy for the edge server and the ECP. Simulation experiments have shown that the solution proposed in this paper leads to a higher utility for the three participants and faster convergence of the algorithm. However, in this paper, we do not discuss how to efficiently and rationally allocate the computational tasks on the ECP, which can be our future work.

**Author Contributions:** Conceptualization and design, F.Z. and Q.D.; experimentation, Q.D.; analysis, F.Z.; reagents/materials/analysis tools, F.Z.; writing, F.Z. and Q.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported in part by the National Science Foundation of China (Grant No. 62172450), the Key R&D Plan of Hunan Province (Grant No. 2022*GK*2008) and the Nature Science Foundation of Hunan Province (Grant No. 2020*JJ*4756).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Yuan, S.; Fan, Y.; Cai, Y. A Survey on Computation Offloading for Vehicular Edge Computing. In Proceedings of the 2019 7th International Conference on Information Technology: IoT and Smart City, Shanghai China, 20–23 December 2019; ACM: Shanghai, China, 2019; pp. 107–112. [CrossRef]
2. Zeng, F.; Chen, Q.; Meng, L.; Wu, J. Volunteer Assisted Collaborative Offloading and Resource Allocation in Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3247–3257. [CrossRef]
3. Zhang, K.; Mao, Y.; Leng, S.; Maharjan, S.; Zhang, Y. Optimal delay constrained offloading for vehicular edge computing networks. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; IEEE: Paris, France, 2017; pp. 1–6. [CrossRef]
4. Li, C.; Wang, S.; Huang, X.; Li, X.; Yu, R.; Zhao, F. Parked Vehicular Computing for Energy-Efficient Internet of Vehicles: A Contract Theoretic Approach. *IEEE Internet Things J.* **2019**, *6*, 6079–6088. [CrossRef]
5. Su, Z.; Xu, Q.; Hui, Y.; Wen, M.; Guo, S. A Game Theoretic Approach to Parked Vehicle Assisted Content Delivery in Vehicular Ad Hoc Networks. *IEEE Trans. Veh. Technol.* **2017**, *66*, 6461–6474. [CrossRef]
6. Du, J.; Yu, F.R.; Chu, X.; Feng, J.; Lu, G. Computation Offloading and Resource Allocation in Vehicular Networks Based on Dual-Side Cost Minimization. *IEEE Trans. Veh. Technol.* **2019**, *68*, 1079–1092. [CrossRef]
7. Ku, Y.J.; Baidya, S.; Dey, S. Adaptive Computation Partitioning and Offloading in Real-Time Sustainable Vehicular Edge Computing. *IEEE Trans. Veh. Technol.* **2021**, *70*, 13221–13237. [CrossRef]
8. Nazar, K.; Saeed, Y.; Ali, A.; Algarni, A.D.; Soliman, N.F.; Ateya, A.A.; Muthanna, M.S.A.; Jamil, F. Towards Intelligent Zone-Based Content Pre-Caching Approach in VANET for Congestion Control. *Sensors* **2022**, *22*, 9157. [CrossRef] [PubMed]
9. Dai, S.; Li Wang, M.; Gao, Z.; Huang, L.; Du, X.; Guizani, M. An Adaptive Computation Offloading Mechanism for Mobile Health Applications. *IEEE Trans. Veh. Technol.* **2020**, *69*, 998–1007. [CrossRef]
10. Li, C.; Liu, F.; Wang, B.; Philip Chen, C.L.; Tang, X.; Jiang, J.; Liu, J. Dependency-Aware Vehicular Task Scheduling Policy for Tracking Service VEC Networks. *IEEE Trans. Intell. Veh.* **2022**, 1–15. [CrossRef]
11. Liu, Y.; Wang, S.; Zhao, Q.; Du, S.; Zhou, A.; Ma, X.; Yang, F. Dependency-Aware Task Scheduling in Vehicular Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 4961–4971. [CrossRef]
12. Luo, Q.; Li, C.; Luan, T.H.; Shi, W. Collaborative Data Scheduling for Vehicular Edge Computing via Deep Reinforcement Learning. *IEEE Internet Things J.* **2020**, *7*, 9637–9650. [CrossRef]

13. Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining Improvements in Deep Reinforcement Learning. *arXiv* **2017**, arXiv:1710.02298.
14. Ning, Z.; Li, Y.; Dong, P.; Wang, X.; Obaidat, M.S.; Hu, X.; Guo, L.; Guo, Y.; Huang, J.; Hu, B. When Deep Reinforcement Learning Meets 5G-Enabled Vehicular Networks: A Distributed Offloading Framework for Traffic Big Data. *IEEE Trans. Ind. Inf.* **2020**, *16*, 1352–1361. [CrossRef]
15. Jang, Y.; Na, J.; Jeong, S.; Kang, J. Energy-Efficient Task Offloading for Vehicular Edge Computing: Joint Optimization of Offloading and Bit Allocation. In Proceedings of the 2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring), Antwerp, Belgium, 25–28 May 2020; IEEE: Antwerp, Belgium, 2020; pp. 1–5. [CrossRef]
16. Zhou, H.; Jiang, K.; Liu, X.; Li, X.; Leung, V.C.M. Deep Reinforcement Learning for Energy-Efficient Computation Offloading in Mobile-Edge Computing. *IEEE Internet Things J.* **2022**, *9*, 1517–1530. [CrossRef]
17. Kumar, A.; Dadheech, P.; Chaudhary, U. Energy Conservation in WSN: A Review of Current Techniques. In Proceedings of the 2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE), Jaipur, India, 7–8 February 2020; IEEE: Jaipur, India, 2020; pp. 1–8. [CrossRef]
18. Shi, J.; Du, J.; Wang, J.; Yuan, J. Distributed V2V Computation Offloading Based on Dynamic Pricing Using Deep Reinforcement Learning. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Republic of Korea, 25–28 May 2020; IEEE: Seoul, Republic of Korea, 2020; pp. 1–6. [CrossRef]
19. Zeng, F.; Chen, Y.; Yao, L.; Wu, J. A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing. *Peer-to-Peer Netw. Appl.* **2021**, *14*, 467–481. [CrossRef]
20. Zeng, F.; Zhang, K.; Wu, L.; Wu, J. Efficient Caching in Vehicular Edge Computing Based on Edge-Cloud Collaboration. *IEEE Trans. Veh. Technol.* **2023**, *72*, 2468–2481. [CrossRef]
21. Li, Y.; Yang, B.; Chen, Z.; Chen, C.; Guan, X. A Contract-Stackelberg Offloading Incentive Mechanism for Vehicular Parked-Edge Computing Networks. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; IEEE: Kuala Lumpur, Malaysia, 2019; pp. 1–5. [CrossRef]
22. Hussain, M.W.; Reddy, K.H.K.; Rodrigues, J.J.P.C.; Roy, D.S. An Indirect Controller-Legacy Switch Forwarding Scheme for Link Discovery in Hybrid SDN. *IEEE Syst. J.* **2021**, *15*, 3142–3149. [CrossRef]
23. Zhang, Z.; Zeng, F. Efficient Task Allocation for Computation Offloading in Vehicular Edge Computing. *IEEE Internet Things J.* **2023**, *10*, 5595–5606. [CrossRef]