Journal of Systems Architecture 121 (2021) 102331

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Optimal computational resource pricing in vehicular edge computing: A Stackelberg game approach

Chaogang Tang [a], Huaming Wu [b,*]

[a] School of Computer Science and Technology, China University of Mining and Technology, 221116, Xuzhou, China
[b] Center for Applied Mathematics, Tianjin University, 300072, Tianjin, China

ARTICLE INFO

Keywords:
Vehicular edge computing
Pricing
Stackelberg game
Privacy
Computational resources

ABSTRACT

Vehicular edge computing (VEC) pushes the computational resources to the logical edge of the networks, thus enabling vehicles to run resource-hungry and time-sensitive applications by outsourcing operations. Many studies revolved around VEC focus on the optimization of response latency, energy consumption, or both of them, assuming that the computational resources in VEC can be utilized for free. However, VEC provisions computational resources on a pay-as-you-go basis, which means VEC can obtain revenues by leasing the computational resources. In this paper, we focus on the real-time computational resource pricing in VEC, in the hope to reach a win–win situation where both VEC and vehicles can optimize their respective utility values. To reach such a mutually satisfactory result, we adopt the Stackelberg game to model the computational resource pricing problem in this paper. In this game, vehicles are followers and the edge server serves as the leader. Furthermore, we have proven that a unique Stackelberg equilibrium exists in the proposed pricing game. A distributed algorithm is put forward to solve our problem, which considers the privacy of vehicles. The distributed approach is evaluated by extensive experiments, in terms of convergence rate, running time and so on. The simulation results demonstrate that the distributed approach can achieve satisfactory results without privacy disclosure compared to the centralized approach.

1. Introduction

Vehicular edge computing (VEC) has generated a vast amount of attention recently for bringing considerable benefits to smart transportation. As a newly emerging computing paradigm, VEC pushes the computational resources to the logical edge of the networks such as road side unit (RSU), thus enabling vehicles to run resource-hungry and time-sensitive applications by outsourcing operations. Compared to vehicle-mounted computers, VEC has much more computational resources. Hence, vehicles can rent these resources close to them, to support resource-greedy applications, e.g., in-car interactive gaming and natural language understanding and processing [1,2]. In contrast to the sensor-to-cloud paradigm where applications are outsourced and performed in a remote cloud center, such a cloud-similar computing paradigm can substantially alleviate traffic congestion in the network core, owing to the computational resources in close proximity to data sources (i.e., moving vehicles) [3]. Accordingly, VEC has become one of the key enablers for smart transportation (e.g., unmanned vehicles).

It shall be noted that a remote cloud center is indispensable in VEC. On one hand, VEC is introduced as an intermediate layer between vehicles and the cloud center with the purpose of extending the computing and storage capabilities at cloud to the network edge to satisfy vehicular applications with strict delay requirements. VEC resorts to the cloud center by renting the computational resources when its own resources are not sufficient. On the other hand, for those vehicular applications which are resource hungry but not time sensitive, cloud computing remains the first choice for application outsourcing, due to the fact that there are unlimited computational and storage resources in the cloud center.

Against this background, many studies revolved around VEC have been carried out in both industry and academia fields, such as [4–7]. Currently, most of these works focus on the optimization of response latency, energy consumption, or both of them, when vehicular applications are offloaded and computational resources are scheduled in VEC. An implying assumption among these works is that the computational resources in VEC or the cloud center can be used for free. As with cloud computing, VEC provisions computational resources on a pay-as-you-go basis. Thus, the resources in either VEC or the cloud center are not for unconditional use in reality. Furthermore, monetary rewards have always been a strong incentive for resource providers in VEC. From the

* Corresponding author.
E-mail address: whming@tju.edu.cn (H. Wu).

https://doi.org/10.1016/j.sysarc.2021.102331
Received 8 May 2021; Received in revised form 28 October 2021; Accepted 1 November 2021
Available online 13 November 2021

perspective of resource providers, they attempt to maximize the revenues by leasing the computational resources in VEC. In contrast, the vehicles strive to accomplish their application outsourcing at the least cost, e.g., with regards to (w.r.t.) response time, energy consumption and monetary expenditure. The pricing for computational resources will play an important role in the maximization of their respective profits. For instance, a higher price for computational resources will bring more revenues to service providers in VEC. In the meanwhile, a lower price for computational resources will encourage vehicles to outsource more vehicular applications and thus the quality of service (QoS) can be improved in terms of response time.

Nevertheless, few existing works have recognized the importance of computational resource pricing for application outsourcing in VEC [8,9]. In this paper, we concentrate on the real-time pricing scheme for computational resources in VEC. In our view, a reasonable pricing scheme is very important, since it not only stimulates more efforts for computational resources contribution in VEC, but also avoids dampening the enthusiasm of vehicles for resource renting. Specifically, the contributions of this paper are summarized as follows.

- We propose an optimal computational resource pricing scheme in VEC. The proposed pricing scheme can guarantee a mutually satisfactory result for vehicles and the edge server. To this end, both vehicles and VEC can optimize respective utility values according to the price per workload set by VEC.
- We adopt the Stackelberg game for real-time computational resource pricing in this paper. Vehicles are followers and the edge server is the leader in this one-leader multi-followers game. Vehicles determine their own workloads to be undertaken by VEC by observing the leader's strategy (i.e., the price per workload). On the other hand, the edge server adjusts the price per workload based on the offloaded workloads of the vehicles. Furthermore, we have proven that a unique Stackelberg equilibrium exists in this pricing game.
- Considering the fact that vehicles generally refuse to disclose their private information to the public, a distributed algorithm is proposed to solve this optimal computational resource pricing issue in VEC. Extensive simulation is conducted to evaluate the performance of our approach, and the simulation results demonstrate the advantages of our approach.

The rest of this paper is organized as follows. We review some related works in Section 2. Section 3 introduces our system model. Our optimization problem is mathematically formulated in Section 4, which strives to optimize the utility values of vehicles and the edge server, respectively. In Section 5, the centralized and distributed algorithms are put forward to solve the computational resource pricing in VEC, respectively. Simulation and result analysis come in Section 6, followed by the conclusion in Section 7.

2. Related works

With the advent of VEC, computing capabilities have been brought to the edge of the network, which brings considerable benefits to smart transportation and the corresponding sub-ecosystems including connected vehicles and RSU. The computational resources are provisioned at the edge to such entities, enabling vehicles to run resource-hungry and time-sensitive applications by outsourcing operations. Accordingly, VEC has been regarded as one of the key enablers to accelerate the prosperity of smart transportation. In this section, we will review some related works in this field.

2.1. Computation offloading in VEC

In smart transportation and VEC systems, it is pretty difficult to predict the routes of moving vehicles [10]. Quick mobility of vehicles and different driver preferences further contribute to such difficulty. To tackle this challenge, Liu et al. in [7] proposed to use one particular type of vehicle to serve as moving servers. Such vehicles (e.g., buses) are usually deployed with a timetable and follow the prescribed route. In this context, they propose an offloading algorithm based on learning technology to perceive the fluctuation of vehicles. Base stations, as agents, are responsible for learning the state of the moving server. Multi-access edge computing is considered to enhance the performance of vehicles by outsourcing computation-intensive tasks to the edge. Thus, authors in [11] put forward an energy-aware computation outsourcing for VEC. They strive to seek a tradeoff between latency and energy consumption.

Parked vehicles usually have unexploited computational resources with idle states in the parking slots. By leveraging these idle computational resources of parked vehicles, authors in [12] aim to maximize user-centric utility and optimize the network-wide task scheduling. Owing to the high dynamics of vehicular networks, it is hard to make time-varying offloading decisions in vehicular networks. Thus, authors in [13] utilized the synchronized random walk model and proposed a reinforcement learning-based scheme for processing delay reduction and dynamic scene adaptability. Similarly, authors in [14] proposed an intelligent task offloading approach using deep Q learning to mitigate the pressure on the computational capabilities of vehicle-mounted computers.

VEC guarantees that computationally intensive workloads can be offloaded to the computing infrastructure in the vicinity. Speak of autonomous vehicles, however, it is very hard for them to efficiently obtain satisfactory performance by leveraging the VEC systems. In this context, authors in [15] proposed a vehicular edge orchestrator based on two-stage machine learning. This orchestrator considers the success rate of task performing and the service time. The simulation results have demonstrated the efficiency and effectiveness of the proposed approach.

Actually, extensive works have focused on computation offloading in VEC for multiple purposes, e.g., energy reduction [16,17], response latency optimization [18–20], trustiness issue [21,22] and the reduction of pressure on the vehicular computational resources. Readers who are interested can refer to the aforementioned works, and we do not review them in detail anymore.

2.2. Resource pricing for computation undertaking

We notice that few of the current works have paid attention to the issue of computational resource pricing in VEC. One possible reason is that few of them have recognized the importance of computational resource pricing for application outsourcing in VEC [23]. The IoT devices usually have numerous tasks and thus urgently require computing resources for undertaking the computation. The computation is usually undertaken based on a pay-as-you-go model. Therefore, the resource pricing will become more and more important with the increasing number of entities that can provide computing resources.

For example, authors in [24] strived to maximize the revenues of the mobile edge computing (MEC) system. The edge server deployed at an access point can provide sufficient computing resources for resource-hungry users. Thus, the edge server can earn revenues by charging users with the task offloading requests. A policy gradient-based reinforcement learning algorithm is put forward to solve this revenue maximization problem. Authors in [25] put forward a market-based framework, which can make full use of the resources of edge nodes for serving the requestors at the network edge. They can generate a market equilibrium solution, i.e., the utility of the edge can be maximized and optimal resources can be allocated to the requestors while considering multiple constraints.

Authors in [26] proposed an edge-intelligent hierarchical dynamic pricing mechanism. In this mechanism, the collaboration among the cloud, edge, and client is modeled as a double-layer Stackelberg game. A pricing prediction algorithm is put forward to solve the problem. A

computing and networking paradigm based on multi-access edge computing was proposed in [27], which tries to cope with the increasingly complicated requirements of Internet of Things users. To model price negotiation between the service providers and the edge nodes, they put forward a distributed algorithm for negotiating the price.

Tasks are also offloaded in multi-access edge computing. Authors in [28] investigated the task offloading and resource allocation in this computing paradigm from the market and economic perspective. They designed an economy-inspired commercial model to realize the resource quota sharing among requestors, in the hope to maximize the overall welfare of requestors. In particular, they designed a distributed pricing mechanism.

2.3. Performance optimization with Stackelberg game for computation undertaking

The Stackelberg game provides a well-suited solution to the multi-level decision-making process, and has been widely applied to scenarios where two entities pursue their respective profits or revenues maximization. For instance, authors in [8] leverage the parked vehicle to assist VEC in undertaking workloads in vehicular networks. Specifically, they studied the assignment of tasks with aid of Stackelberg game, for minimizing the overall costs. A price-based distributed approach is put forward in [29], with the purpose of managing the tasks offloaded. Specifically, a Stackelberg game is applied to the situation for realizing the respective profits from the viewpoints of the edge and users.

IoT devices require a dependable environment for performance guarantee and the blockchain technology can provide a promising solution to the requirement. However, the blockchain tasks are featured by intensive computation while these IoT devices have limited computing resources. Thus, task offloading is necessary for these IoT devices. Usually, the cloud center and edge servers are the places to undertake these offloaded computations. Therefore, authors in [30] state that the costs and profits of computing resources providers can greatly affect the decisions of the task allocation. Specifically, they formulate a Stackelberg game where the cloud center and edge servers are the leader and the followers in the computing resource management, respectively. Furthermore, they model resource pricing as a mixed-integer programming problem.

Different from the aforementioned works, in this paper we focus on pricing per workload which is undertaken by VEC with strict latency requirements in VEC. Vehicles can determine the number of their workloads to be offloaded given the unit price. In the meanwhile, the privacy of vehicles are considered in this paper.

3. System model

An application scenario is shown in Fig. 1, which consists of one RSU, one remote cloud center and multiple vehicles. The edge server is deployed at RSU to provide computational resources to the vehicles in the vicinity. The set of vehicles is denoted by $V = \{v_1, \ldots, v_n\}$. Each vehicle in V can communicate with each other and RSU, using Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communication technologies, respectively. The maximal communication distance of RSU is D as shown in Fig. 1. As a result, the dwell time for any v_i in V within the communication range of RSU is limited. In this context, assume there exists a set of applications denoted by $A = \{a_1, \ldots, a_n\}$ with $a_i (1 \le i \le n)$ denoting the application that v_i wants to outsource to the edge server for execution. Specifically, a_i is a 2-tuple of $(d_i, w_{i,max})$, where d_i represents the size of task-input data to be offloaded via the wireless channel, and $w_{i,max}$ is the workload described by the number of CPU cycles required for accomplishing a_i when a_i is totally outsourced to the edge.

Let p denote the unit price for processing per workload in VEC. Usually, the price p is set by the resource provider in VEC. Considering the

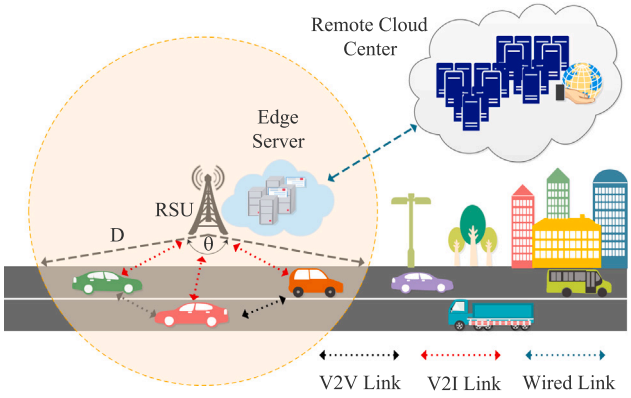

Fig. 1. Application scenario considered in this paper.

effect of p on the computing resource demands of vehicle v_i, we have $w_i \le w_{i,max}$, where w_i represents the real workload that is offloaded by v_i to the edge server. $w_{i,max} - w_i$ is the residual workload that needs to be accomplished with the local vehicular computational resources. It shall be noted that $w_{i,max} - w_i$ actually violates the willingness of v_i, since the residual workloads are supposed to be kept for any other individual purposes. Thus, to express such unwillingness, we define a subjective dissatisfaction as $\delta(w_{i,max} - w_i)$, where δ is a controlling factor that reflects the level of dissatisfaction for the vehicles. For simplicity, we assume that δ is the same for all the vehicles and is also available to RSU. For instance, δ can be incorporated into the beacon information when vehicles interact with RSU. Moreover, for easy discussion, hereafter we use the three words "RSU", "the edge server" and "the edge" exchangeably for the same meaning through the rest of the paper.

The response time for performing a_i mainly includes three parts, i.e., the time taken to perform $w_{i,max} - w_i$ locally, the time taken to offload d_i to VEC, and the time taken to perform w_i at RSU. Denote the three kinds of time by t_i^{loc}, t_i^{off} and t_i^e, respectively. t_i^{loc} can be given as:

$$t_i^{loc} = \frac{w_{i,max} - w_i}{f_i}, \tag{1}$$

where f_i denotes the computing capability of v_i. Denote by g_i and p_i the channel gain between v_i and RSU, and the transmission power of v_i, respectively. The offloading rate for a_i can be calculated as:

$$r_i = B \log_2(1 + \frac{p_i g_i}{\sigma^2}), \tag{2}$$

where B is the channel bandwidth and σ^2 is the noise power.

Thus, the offloading time t_i^{off} is:

$$t_i^{off} = \frac{d_i}{r_i}. \tag{3}$$

The execution time of a_i at the edge can be expressed as:

$$t_i^e = t_{init} + \frac{S_i}{f_e}, \tag{4}$$

where t_{init} is the time taken to initialize the virtual environment and f_e denotes the computing capability of RSU. For simplicity, we assume there are sufficient computational resources at RSU, since RSU can also rent these resources from the cloud center. Therefore, there is no extra queuing time for the arrived applications. On the other hand, following other works such as [31,32], we neglect the return time, i.e., the time taken to return the execution result back to v_i, based on the consensus that the size of computational result is much smaller than that of task-input data in most cases. Therefore, the response time for a_i can be expressed as:

$$t_i^{rt} = \max\{t_i^{loc}, t_i^{off} + t_i^e\}. \tag{5}$$

On the other hand, the dwell time of v_i within the communication range of RSU can be estimated as:

$$dt_i = \frac{2D\sin(\theta/2)}{r_i}, \tag{6}$$

where θ is the angle formed between R and the points where v_i enters and leaves the communication range of R, respectively, as shown in Fig. 1.

The application a_i should be completed before v_i leaves the communication range of R, so the following inequality holds:

$$t_i^{rt} \le dt_i, \quad \forall i \in \{1, \dots, n\}. \tag{7}$$

Based on the descriptions above, the utility of v_i can be defined as:

$$\mathcal{U}_i = \alpha_i \log(1 + w_i) + \delta(w_{i,max} - w_i) - p \cdot w_i, \tag{8}$$

where α_i is the satisfaction level toward the workload offloading. For the right-hand side of this equation, the first term represents the utility earned by offloading workload w_i using a logistic function, the second term evaluates the unwillingness of using local computational resources to undertake the residual workload, and the last term denotes the payments for renting resources from VEC. Generally, the larger the value of $\mathcal{U}_i$, the higher the level of satisfaction for vehicles.

On the other hand, the computational resource provider in VEC is encouraged to pursue their own profits by leasing the resources to vehicles in the vicinity. It shall be noted that VEC occasionally turns to the cloud center for help, if the amount of computational resources in VEC is not sufficient. Accordingly, there exists a certain cost for VEC to provide computational resources. Given the unit price p for processing per workload offloaded from vehicles, the utility of the resource provider in VEC can be defined as:

$$\mathcal{U}_e = p \sum_{i=1}^{n} w_i - \eta \sum_{i=1}^{n} w_i, \tag{9}$$

where η denotes the cost for undertaking per workload offloaded from the vehicles, the first term at the right-hand side is the profits obtained by leasing computing resources and the second term denotes the costs when VEC provide computational resources to vehicles. Generally, the larger the value of $\mathcal{U}_e$, the more the revenues for VEC.

4. Problem formulation

4.1. Preliminaries

According to the above descriptions, we hope that (1) vehicles can determine their own workloads to be offloaded based on the price per workload that is set by VEC, such that they can optimize their own utility values; (2) VEC can maximize its own revenues by pricing the computational resources while considering the real demands of vehicles in the vicinity for computational resources. Therefore, it is very important to design an efficient and reasonable pricing scheme for computational resources in VEC, for the reason that an appropriate price p can help both vehicles and RSU reach a mutually satisfactory result, i.e., utility optimization for vehicles and revenues maximization for VEC, respectively. However, it is very challenging to reach such a win-win situation in which both vehicles and RSU can achieve their own goals for the following reasons.

On one hand, vehicles are selfish in the sense that they are not willing to disclose their true resource demands to other vehicles except RSU. From the perspective of privacy protection [33], vehicles are not supposed to reveal their resource demands to other vehicles either. In this context, vehicles individually adjust their resource demands based on the price towards their own utility maximization. No information on resource demands (e.g., the satisfaction level α_i of v_i) is shared among these vehicles. Therefore, a centralized algorithm is inapplicable in this case, which necessitates a distributed approach for utility optimization.

On the other hand, the high mobility of vehicles poses strict delay requirements to application outsourcing. For instance, the response time of an outsourced application should be strictly less than the corresponding dwell time as denoted by Eq. (7). The price per workload should be determined before the application outsourcing actually takes place. As a consequence, the computational resource pricing is supposed to be real-time, which contributes to the difficulty of interactions between vehicles and RSU.

Before going further, the dynamic procedure of computational resource pricing can be briefly described below. In the beginning, a price per workload p is initialized and then broadcasted by RSU to the vehicles in the vicinity. Such information can be incorporated into the beacon information and disseminated among vehicles. Upon receiving these beacons, vehicles with the outsourcing need to estimate the number of workloads to be undertaken by VEC, according to p. Then, the estimated amount of workloads is respectively sent to RSU by each vehicle during the beacon information exchanging. After receiving these demands, the edge will evaluate and determine a new p such that its revenues can gradually increase. The resulting p is then broadcasted to vehicles again where the amount of workloads to be undertaken is recalculated based on p, with an aim to gradually increase their own utility values. Such a procedure will not stop until a mutually satisfactory result is achieved, e.g., neither the revenues of VEC nor the utility values of vehicles increase anymore.

4.2. Stackelberg game

In this paper, we resort to the Stackelberg game for modeling this dynamic procedure of computational resource pricing as described above. As introduced earlier, the Stackelberg game offers a promising solution to the multilevel decision-making process consisting of the followers and the leader. At the beginning, the followers observe the leader's strategy. Then, they select their own strategies for the sake of their own utility optimization. In the next, they respond to the leader by the chosen strategies. Finally, the leader redesigns the strategy in response to the followers. This procedure repeats until a so-called Stackelberg equilibrium is achieved if the equilibrium exists. Accordingly, we utilize this one-leader multi-followers Stackelberg game to solve our computational resource pricing scheme for VEC in this paper. In this noncooperative game, RSU serves as the leader and vehicles are the followers. To be specific, the game of the computational resource pricing scheme in VEC is given as:

$$\mathcal{G} = (V \cup RSU, (p, w_i), (\mathcal{U}_e\{w_i\}, \mathcal{U}_i\{p\})), \tag{10}$$

where $V \cup RSU$ is the set of players, including the followers (i.e., the set of vehicles) and the game leader (i.e., RSU), (p, w_i) denotes the set of strategies VEC and vehicles take respectively, and $(\mathcal{U}_e\{w_i\}, \mathcal{U}_i\{p\})$ denotes the set of utility values of VEC and vehicles, respectively. Generally, the Stackelberg consists of the set of players, the set of strategies and the utility set in this paper.

In the next section, we mathematically formulate the optimization problem from the viewpoints of vehicles and RSU, respectively.

4.3. Vehicle utility optimization

Given the price per workload p, vehicles with the outsourcing needs strive to optimize their own utility values by adjusting their workloads to be undertaken by VEC. Accordingly, the optimization problem for each vehicle can be mathematically given as:

$$P1: \quad \max_{p} \quad \mathcal{U}_i \tag{11}$$

s.t.:

$$w_i \le w_{i,max}, \quad \forall i \in \{1, \dots, n\}, \tag{12}$$

$$t_i^{rt} \le dt_i, \quad \forall i \in \{1, \dots, n\}, \tag{13}$$

where in Eq. (11) represents that the assigned workloads to VEC w_i should not exceed $w_{i,max}$. In the meanwhile, the response time for the workload w_i should not exceed the dwell time of v_i within the communication range of RSU, which can be guaranteed by the constraint (13).

Suppose that the offloaded workloads w_i is continuous. For the utility function of vehicle v_i, i.e., $\mathcal{U}_i$, take the first and second derivatives of $\mathcal{U}_i$ with respect to w_i respectively, we can obtain:

$$\frac{\partial \mathcal{U}_i}{\partial w_i} = \frac{\alpha_i}{(1+w_i)\ln 2} - \delta - p, \tag{14}$$

$$\frac{\partial^2 \mathcal{U}_i}{\partial w_i^2} = \frac{-\alpha_i}{(1+w_i)^2 \ln 2} < 0. \tag{15}$$

Due to the fact that the second derivative of $\mathcal{U}_i$ is always negative in the feasible domain, the utility function $\mathcal{U}_i$ is thus convex in terms of the workloads w_i. As a result, problem $P1$ is a convex optimization problem. We can easily infer that the maximum of $\mathcal{U}_i$ exists, and the corresponding maximal value can be obtained by making $\partial \mathcal{U}_i / \partial w_i = 0$ hold, given below:

$$w_i^* = \frac{\alpha_i}{(p+\delta)\ln 2} - 1. \tag{16}$$

It shall be noted that α_i is private to each vehicle v_i, so v_i does not want to disclose it to other vehicles for the sake of privacy protection. On the other hand, the value of α_i should be appropriately set such that $w_i > 0$ holds. It can be easily observed that the workloads w_i to be undertaken by VEC decreases with the increasing p. Namely, the price p should be set reasonably, since higher pricing for the computational resources will hold back the enthusiasm of vehicles with outsourcing needs in the vicinity.

4.4. Revenue maximization for VEC

It is costly to deploy the edge server together with RSU in the traffic dense area, let alone daily maintenance costs. In this context, resource providers are encouraged to contribute their resources to VEC. On one hand, this way benefits vehicles in the vicinity a lot since vehicular applications can be offloaded and executed in VEC instead of a cloud center, which can drastically reduces the response time of the outsourced applications. On the other hand, resource provisioning in a pay-as-you-go model will stimulate providers to lease computational resources with QoS satisfaction. Therefore, the purpose of VEC (i.e., RSU) in this paper is to maximize its revenues defined in Eq. (9). To be more specific, the optimization problem for RSU is mathematically given as:

$$P2: \quad \max_{\boldsymbol{w}} \quad \mathcal{U}_e \tag{17}$$

s.t.

$$p > \eta, \tag{18}$$

where $\boldsymbol{w} = (w_1, \ldots, w_n)$ is a vector to denote the workloads of all the vehicles which are about to be undertaken by VEC. Constraint (18) guarantees that the revenues of VEC are positive, i.e., it is profitable for providers to lease the computational resources in VEC.

Let us suppose that p is a continuous variable and take the first derivative of $\mathcal{U}_e$ in terms of p, i.e.,

$$\frac{\partial \mathcal{U}_e}{\partial p} = \frac{\partial (p \sum_{i=1}^n w_i - \eta \sum_{i=1}^n w_i)}{\partial p}. \tag{19}$$

Substitute w_i with p as shown in Eq. (16), and we have

$$\partial \mathcal{U}_e = \partial \left[p \sum_{i=1}^n \frac{\alpha_i}{(p+\delta)\ln 2} - \eta \sum_{i=1}^n \frac{\alpha_i}{(p+\delta)\ln 2} - (p-\eta)n \right]. \tag{20}$$

Let $\partial \mathcal{U}_e / \partial p = 0$, namely,

$$\frac{\partial \mathcal{U}_e}{\partial p} = \sum_{i=1}^n \frac{\alpha_i(\delta+\eta)}{(p+\delta)^2 \ln 2} - n = 0. \tag{21}$$

Thus, the price p^* can be calculated as:

$$p^* = \sqrt{\frac{\delta+\eta}{n \ln 2} \sum_{i=1}^n \alpha_i} - \delta. \tag{22}$$

Take the second of derivative of $\mathcal{U}_e$ with respect to p, we have:

$$\frac{\partial^2 \mathcal{U}_e}{\partial p^2} = \frac{\partial (\sum_{i=1}^n \frac{\alpha_i(\delta+\eta)}{(p+\delta)^2 \ln 2} - n)}{\partial p} = -\sum_{i=1}^n \frac{2\alpha_i(\delta+\eta)}{(p+\delta)^3 \ln 2} < 0. \tag{23}$$

Due to the fact that the second derivative of $\mathcal{U}_e$ is always negative in the feasible domain, the utility function $\mathcal{U}_e$ is thus convex in terms of the workloads p. Therefore, problem $P2$ is a convex optimization problem. We can infer that p^* is the best price that can maximize $\mathcal{U}_e$. In the meanwhile, it can also be easily observed that the optimal price p^* totally depends upon the number of vehicles n, the cost for undertaking per workload η, and the satisfaction level toward the outsourced workload for each vehicle α_i. Accordingly, the optimal price p^* is actually independent of the workload vector $\boldsymbol{w}$.

4.5. Stackelberg equilibrium

$P1$ and $P2$ are the optimization goals of vehicles and RSU, respectively. Let us investigate the optimization in depth from the game-theoretical perspective. The Stackelberg game proposed for pricing the computational resources in this paper needs to find the Stackelberg equilibrium (SE), since such a Stackelberg equilibrium can guarantee that a mutually satisfactory result can be reached for both vehicles and VEC, i.e., utility optimization for vehicles and revenues maximization for VEC. In this section, we will prove the existence and uniqueness of the Stackelberg Equilibrium (SE). By doing this, we can ensure that RSU can obtain the optimal resource pricing and vehicles can determine the optimal amount of workloads to be undertaken by VEC.

Let $\boldsymbol{w} = \{w_1, w_2, \ldots, w_n\}$ and $\boldsymbol{p}$ denote the strategies of vehicles and RSU, respectively. Then we define the SE of $\mathcal{G}$ as below:

Definition 1. Assume there exists a pair of strategy profiles $(\boldsymbol{w}^*, p^*)$ for $\mathcal{G}$ and $\boldsymbol{w}^* = \{w_1^*, w_2^*, \ldots, w_n^*\}$. For an arbitrary vehicle $v_i (\in V)$, it cannot increase its own utility value anymore by adjusting the strategy unilaterally, i.e.,

$$\mathcal{U}_i(w_i^*, w_{-i}^*, p^*) \geq \mathcal{U}_i(w_i', w_{-i}^*, p^*), \quad \forall i \in \{1, 2, \ldots, n\}, \tag{24}$$

where $w_{-i}^* = \{w_1^*, \ldots, w_{i-1}^*, w_{i+1}^* \cdots, w_n^*\}$ denotes the strategies of other vehicles except v_i and w_i' is an arbitrary strategy of v_i except w_i^*. In the meanwhile, RSU cannot obtain more revenues by adjusting the price p either, i.e.,

$$\mathcal{U}_e(w_i^*, w_{-i}^*, p^*) \geq \mathcal{U}_e(w_i^*, w_{-i}^*, p'), \quad \forall i \in \{1, 2, \ldots, n\}. \tag{25}$$

In this case, we call this pair of strategy profiles $(\boldsymbol{w}^*, p^*)$ is the SE of $\mathcal{G}$.

It is worthwhile mentioning that equilibrium does not always exist in a non-cooperative Stackelberg game. As a result, it is necessary to prove the existence and uniqueness of SE for $\mathcal{G}$ proposed in this paper.

Theorem 1. *A unique SE always exists in the proposed Stackelberg game* $\mathcal{G} = (V \cup RSU, (p, w_i), (\mathcal{U}_e\{w_i\}, \mathcal{U}_i\{p\}))$ *for computational resource pricing in VEC.*

Proof. Existence: In this Stackelberg game, the vehicles (i.e., the followers in $\mathcal{G}$) are obviously independent of each other. According to Eq. (8), the utility values of them only depend upon the price per workload p and their own amount of workloads that are assigned to VEC. We have proven earlier that the utility function $\mathcal{U}_i$ is strictly convex, for the reason that $\partial^2 \mathcal{U}_i / \partial w_i^2 < 0$ always holds, for an arbitrary vehicle v_i. As a result, after receiving a price p broadcasted by VEC, vehicle v_i can adjust the workloads to be undertaken by VEC with an aim to maximize its utility value based on this price. The workloads

w_i for vehicle v_i vary between the interval $[w_{i,min}, w_{i,max}]$, with an assumption that $w_{i,min}$ is the minimal workloads for v_i to offload. Therefore, there only exist three candidate workloads that need to be examined for the sake of the utility function maximization – $w_{i,min}$, $w_{i,max}$, and $\alpha_i/((p+\delta)\ln 2) - 1$, respectively. According to Definition 1, $\mathcal{G}$ can reach SE if and only if an arbitrary vehicle v_i and RSU obtain the maximal utility and revenues, respectively. Therefore, the game $\mathcal{G}$ can reach the SE as long as RSU finds the optimal price p^* and v_i determines an optimal amount of workloads to offload based on p^*.

As shown in Eq. (23), the second derivative of $\mathcal{U}_e$ w.r.t. p is negative, which makes sure that $\mathcal{U}_e$ is strictly convex in the feasible domain w.r.t. p. Hence, the optimal price per workload p^* can be obtained as shown in Eq. (22). Furthermore, this optimal price is proven to be dependent upon the number of vehicles n, the cost for undertaking per workload η, and the satisfaction level of each vehicle. It is thus totally independent of $\boldsymbol{w}$. As mentioned above, the optimal workload vector $\boldsymbol{w}^*$ can be calculated based on p^*. Accordingly, there always exists a unique SE in $\mathcal{G}$.

Uniqueness: It is worthwhile mentioning that the optimal price p^* obtained in Eq. (22) is unique, for the reason that the other value $-\sqrt{(\delta+\eta)/(n\ln 2)\sum_{i=1}^{n}\alpha_i} - \delta(<0)$ is not qualified as the price per workload. Thus, the workload vector $\boldsymbol{w}*$ is also unique. Accordingly, the SE of $\mathcal{G}$ is unique in this paper. □

5. Algorithm design

An efficient algorithm is needed for optimal computational resource pricing in VEC. First, both vehicles and VEC are satisfied with the computational resource pricing. In other words, the algorithm must be ensured to converge to the unique equilibrium of the proposed Stackelberg game in this paper. Only at this particular time can a win-win situation be reached. Second, the high mobility of vehicles has posed strict delay requirements to the response time when applications are outsourced. The computational price per workload should be determined in almost real time in the stage of the beacon information exchanging. That is to say, the remaining time within the communication range of RSU should be enough for the accomplishment of application outsourcing. Accordingly, the proposed algorithm should be of a fast convergence rate. Last but not least, there is an increasingly urgent need for privacy protection in VEC. For instance, vehicles may be unwilling to disclose their private information (e.g., the satisfaction level and even the utility function) to the public (e.g., other vehicles and RSU). It is challenging to design an efficient algorithm while considering the privacy of each vehicle.

To tackle the above practical considerations, we propose centralized and distributed algorithms respectively in the following two subsections.

5.1. Centralized algorithm

A centralized algorithm (CA) is proposed for pricing the computational resource in VEC as shown in Algorithm 1. It shall be noted that CA does not need the frequent interactions between vehicles and RSU, and thus can be implemented in real time. In Algorithm 1, CA needs not only the number of vehicles, but also the satisfaction levels of vehicles. As shown in lines 1–4 in this algorithm, each vehicle sends its own satisfaction level to RSU together with the beacon information. After receiving all the information from the vehicles, RSU can immediately calculate the optimal price per workload p^* (line 5) and then sent it to all the vehicles. Upon receiving the optimal price, each vehicle v_i can calculate its best workloads to be undertaken by VEC w_i^* from the three candidate workloads, i.e., $w_{i,min}, w_{i,max}$, and $\alpha_i/((p+\delta)\ln 2) - 1$ (line 7). Furthermore, the optimal utility value $\mathcal{U}_i^*$ can also be obtained based on Eq. (8). Finally, the optimal workload profile $\boldsymbol{w}^*$ and price p^* can be returned.

Algorithm 1: Centralized Algorithm for Per-workload Pricing (CAPP)

Input: n, η, δ
Output: The optimal price p^* and $\boldsymbol{w}^*$
1 **for** each v_i in V **do**
2 v_i sends the satisfaction level α_i to RSU together with beacon information;
3 RSU receives and records α_i for v_i;
4 **end**
5 RSU calculates the optimal p^* based on Eq. (22): $p^* = \sqrt{\frac{\delta+\eta}{n\ln 2}\sum_{i=1}^{n}\alpha_i} - \delta$;
6 RSU sends p^* to n vehicles;
7 v_i determines the best workload w_i^* from the following candidates: $\{w_{i,min}, w_{i,max}, \alpha_i/((p+\delta)\ln 2) - 1\}$;
8 Each vehicle v_i calculates its own utility value by: $\mathcal{U}_i^* = \alpha_i \log(1+w_i^*) + \delta(w_{i,max} - w_i^*) - p^* \cdot w_i^*$;
9 v_i sends w_i^* back to RSU;
10 return p^* and $\boldsymbol{w}^*$;

Complexity analysis. As discussed earlier, the time is mainly spent by CS in collecting the information of vehicles such as the satisfaction level (lines 1–4), which is of time complexity $O(n)$. Then, RSU calculates the optimal p^* with constant time. Then, the optimal price p^* is sent to all the vehicles, which requires the time complexity $O(n)$ (line 6). After determining the most suitable workloads to be offloaded, each vehicle sends its respective workloads to RSU, which is of time complexity $O(n)$ (line 9). To sum up, the total time complexity is $O(n)$ for the CS strategy.

However, an implicit assumption in CA is made that all the vehicles are willing to disclose to RSU their private information such as the satisfaction levels and the utility function. Only in this way, the optimal price p^* can be calculated, for the reason that the substitution of w_i with p during the calculation of the first derivative of $\mathcal{U}_e$ w.r.t. p requires the utility function of v_i. From the perspective of privacy protection, such private information is not supposed to be revealed to the public. In this context, it is very necessary to design a distributed algorithm for computational resource pricing in VEC while taking into consideration the privacy protection from the perspective of vehicles.

5.2. Distributed algorithm

Considering the fact that vehicles may not be willing to expose private information to RSU, a distributed algorithm is therefore proposed in this section for obtaining the optimal price. The purpose of our pricing game $\mathcal{G}$ is to reach the SE, which should be guaranteed in the distributed algorithm. To that end, frequent interactions between vehicles and RSU are required for alternating negotiations.

Generally, the procedure of alternating negotiations between vehicles and RSU can be sketched out as follows. First, a price per workload is initialized randomly by RSU and then is sent to each vehicle with the outsourcing needs. Upon receiving the price p, each vehicle begins to compute its own workloads w_i and forwards it to RSU, respectively. A workload vector $\boldsymbol{w}$ is constructed at the edge server, i.e., $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$. According to $\boldsymbol{w}$, the new revenue can be calculated using Eq. (9) at RSU. If the revenue is better than in the past, it means that the price can be further increased strategically, for the purpose of profit maximization. The updated price is then sent back to the vehicles where vehicles leverage it to determine new workloads to be undertaken in VEC.

The above procedure continues until the vehicles and RSU iteratively reach the SE of $\mathcal{G}$ in a distributed way, as defined in Definition 1. Specifically, the corresponding two algorithms, one for vehicles side and the other for RSU side, are shown in Algorithm 2 and Algorithm 3, respectively. Given the price p, the process of determining the optimal

Algorithm 2: Distributed Algorithm for Determining Workloads at Vehicles (DADW)

Input: p, δ, η
Output: The optimal workload w^*

1 **for** each v_i in V **do**
  // Initialize the best utility value for each vehicle
2 $\mathcal{U}_i^* = 0$;
3 **end**
  // Vehicles execute the following codes in a distributed way
4 **for** each iterative p received from RSU **do**
5 **for** each v_i in V **do**
6 $w_i' = \alpha_i/((p+\delta)\ln 2) - 1$;
7 Given w_i', compute t_i^{rt} based on Eq. (5);
8 Compute the dwell time dt_i based on Eq. (6);
9 **if** $t_i^{rt} \le dt_i$ **then**
10 $w_i^* = w_i'$;
11 **if** $w_i' < w_{i,min}$ **then**
12 $w_i^* = w_{i,min}$;
13 **else**
14 **if** $w_i' > w_{i,max}$ **then**
15 $w_i^* = w_{i,max}$;
16 **end**
17 **end**
18 **end**
19 Vehicle v_i calculates its utility by: $\mathcal{U}_i = \alpha_i \log(1 + w_i^*) + \delta(w_{i,max} - w_i^*) - p \cdot w_i^*$;
20 **if** $\mathcal{U}_i > \mathcal{U}_i^*$ **then**
21 $\mathcal{U}_i^* = \mathcal{U}_i$;
22 **end**
23 v_i sends w_i^* back to RSU;
24 return w_i^*;
25 **end**
26 **end**

workload to be undertaken by RSU is shown in Algorithm 2. In the beginning, the best utility for each vehicle is initially set to zero (lines 1–3). The currently optimal workload can be obtained according to Eq. (16). Then the dwell time of the vehicle can be calculated based on this workload. If the dwell time does not exceed the deadline, then DADW checks whether the current workload is valid (lines 11–17) and updates the optimal workload in case of invalidity. The vehicle then calculates its own utility value. Finally, the current vehicle reports the currently optimal workload to RSU.

On the other hand, DAPW is responsible for describing the actions of RSU after receiving the workloads from all the vehicles (i.e., $\boldsymbol{w}$). In the beginning, DAPW initializes the globally best utility value (i.e., the profits) and the per-workload price, respectively (lines 1–2). After that, RSU updates its best utility as long as the workload vector $\boldsymbol{w}$ is constructed. It is noticeable that the utility function $\mathcal{U}_e$ is of the monotonic feature w.r.t. p. Therefore, from the perspective of VEC, the more the pricing, the greater the benefits. Then, DAPW updates p gradually. For instance, p is increased with an increment Δp each time. The resulting p is then sent back to the vehicles. This procedure repeats until the best utility value does not change anymore.

Complexity Analysis. To reach a mutually satisfactory result, frequent interactions for negotiation are required between vehicles and RSU, which means DADW and DAPW are respectively executed by vehicles and RSU in an alternating fashion. It is obvious that both DADW and DAPW have constant-time complexity in theory. Furthermore, each vehicle independently performs DADW, and DAPW is performed at RSU with rich computational resources. These factors can guarantee that the

Algorithm 3: Distributed Algorithm for Pricing Per Workload at RSU (DAPW)

Input: n, η, δ, $\boldsymbol{w}$
Output: The optimal price per workload p^*

1 $\mathcal{U}_e^* = 0$;
2 $p^* = 0$;
3 $s = 0$;
4 **for** each element w_i in $\boldsymbol{w}$ **do**
5 $s += w_i$;
6 **end**
7 $\mathcal{U}_e = (p - \eta) \cdot s$;
8 **if** $\mathcal{U}_e - \mathcal{U}_e^* > \epsilon$ **then**
9 $\mathcal{U}_e^* = \mathcal{U}_e$;
10 $p^* = p$;
11 Update p based on given strategy;
12 Send p to each v_i;
13 **end**
14 return p^*;

two algorithms can be accomplished in real time. As a result, the time is mainly spent on the interactions for negotiation, with the purpose of reaching SE of the game. During this procedure, the update of step increment (i.e., Δp) will have a great effect on the rate of convergence to SE, this is because the convergence rate can be reduced when Δp is small, and on the other hand, the convergence point can be missed when Δp is large. However, it is pretty hard to determine the optimal value of Δp. Accordingly, in this paper the value of Δp is mainly set empirically and we will investigate it further in the next section.

6. Numerical results

The simulation results are reported and analyzed in this section. Before going further, we have listed some key parameters to be used, as denoted in Table 1. For instance, the number of vehicles in the simulation is set to be 20. The communication range D and the angle θ is 100 meters and 60 degrees, respectively. Thus, the dwell time of each vehicle can be estimated. The unit cost η and controlling factor δ are set to be 1 and 0.2, respectively. The satisfaction level ranges from 500 to 1000 while δ is 0.2 for all the vehicles. In the meanwhile, the increment to the price Δp is set to 0.05 as the default value. On another hand, all the simulation is run on a notebook with a 1.8 GHz Intel(R) Core(TM) i5-8250U CPU, 8 GB of RAM, Microsoft Windows 10 Operating System, Python 3.7. The data involved in this simulation is generated empirically, following the previous works such as [23,34].

On one hand, we compare the approach with the benchmark approach in terms of efficiency and effectiveness. In particular, the centralized approach CAPP is adopted as the benchmark algorithm in the experiment. Since the benchmark approach can directly obtain the optimal price for computing resources and the optimal workloads for vehicles to offload, it avoids the frequent interactions between vehicles and RSU. On the other hand, as far as the distributed approach itself is concerned, several factors may affect its performance. Such factors include the step and the number of applications which are offloaded. Accordingly, we also need to evaluate the effects of these factors upon the performance of our approach.

In the experiment, the evaluation metrics for efficiency and effectiveness mainly include the utility value, optimal price, running time and optimal workloads. By comparing these performance indicators between the benchmark algorithm and the proposed approach, we can investigate the feasibility of our approach. The metrics for the involved parameters evaluation include the running time and utility values. By running time, we mean the time taken to reach the SE of the game.

Table 1
Parameter settings.

Notations	Meanings	Default values
n	Number of vehicles	20
D	Communication range	100
θ	Angle	60
η	The unit cost	1
δ	The control factor	0.2
$w_{i,min}$	The minimal workloads undertaken by VEC	0
$w_{i,max}$	The maximal workloads undertaken by VEC	[40, 600]
α_i	The satisfaction level	[500, 1000]
Δp	The increment to price p	0.05

6.1. Performance evaluation for vehicle side

The SE of the real-time pricing game can be reached by iterative interactions between vehicles and RSU. The vehicles update the strategies, i.e., report to RSU their new workloads to be offloaded after observing the new price per workload. The vehicles seek to find the optimal workloads based on Eq. (16), such that their own utility values can be optimized. The experiments are thus conducted to investigate the performance of the proposed approach from the viewpoint of vehicles. The experimental results are shown in the following. Specifically, the workloads vary with the increasing number of interactions and the result is shown in Fig. 2. In this experiment, the number of vehicles is set to 20, i.e., the number of applications (n) is also 20. We randomly take four from these vehicles as the observation subjects. It shall be noted that each iteration corresponds to one unique price per workload. Thus, the figure also demonstrates workload variations with the increasing number of prices.

It can be easily observed that each vehicle tends to decrease its own workloads to be offloaded when the price per workload increases. However, the decrement in the amount of workloads will stop when the SE of the pricing game is reached. For example, the number of workloads to be offloaded does not change anymore when the number of iterations comes to 220 in Fig. 2. On the other hand, the variation of utility value for each vehicle with the increasing number of iterations is shown in Fig. 3. Similarly, each iteration also uniquely corresponds to one price per workload, and this figure also demonstrates the utility variations with the increasing price per workload.

It is noticeable that the utility values of the four vehicles decrease with the increasing number of iterations. It is a procedure of gaming between vehicles and RSU. The initial price p is set to 25 for vehicles empirically. However, from the viewpoint of RSU, the current p may not be a satisfactory price, since the revenues obtained by leasing computational resources can still increase. Thus, RSU as the game leader gradually increases the price in the game and vehicles follow the strategy by decreasing their workloads, until the SE of the game arrives. Accordingly, the utility values of these vehicles also decrease with the increasing number of iterations until the arrival of the SE of the game.

The resource pricing game between vehicles and RUS usually takes place at the beacon exchanging stage. After that, computations are offloaded and undertaken by RSU. Therefore, the time taken to reach the SE of game is supposed to be very soon. The dwell time of vehicles within the coverage of RSU is limited. The application outsourcing should be accomplished during the dwell time, not to mention the decision-making for resource price. A set of experiments has been conducted to investigate the time that is taken to reach the SE of game. In particular, the average running time for one vehicle to reach SE of the game is shown in Fig. 4. Based on the observations shown in Figs. 2 and 3, the SE of the game can be reached with the number of iterations equal to 220. As denoted in Fig. 4, it takes about 220 ms to reach the SE of the game. The time is almost real time and thus acceptable for us, especially considering our simulation settings. We believe that it will take much less time for RSU with more powerful capabilities to reach the SE of game.

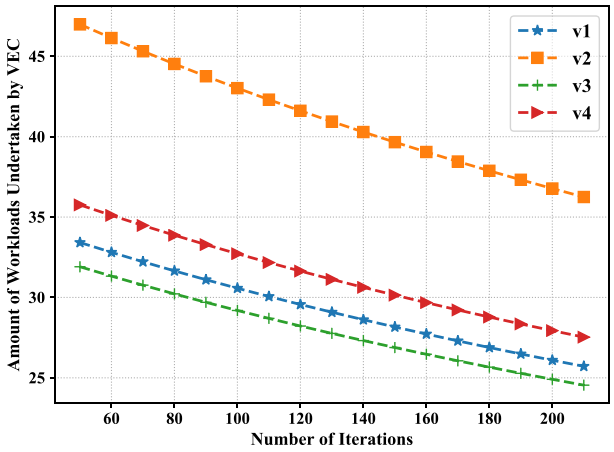

Fig. 2. The workload variations with the increasing number of interactions.

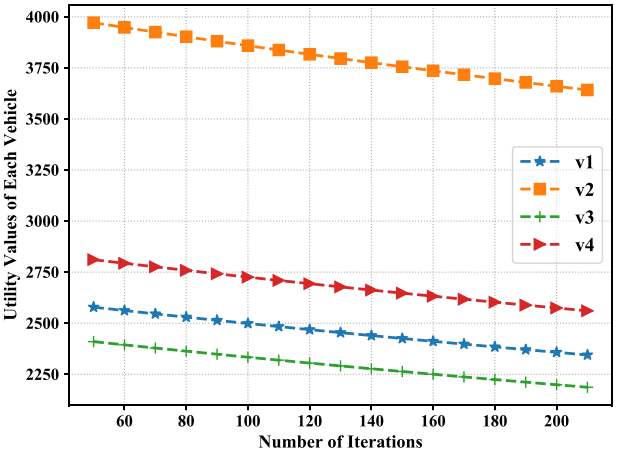

Fig. 3. The utility variations with the increasing number of interactions.

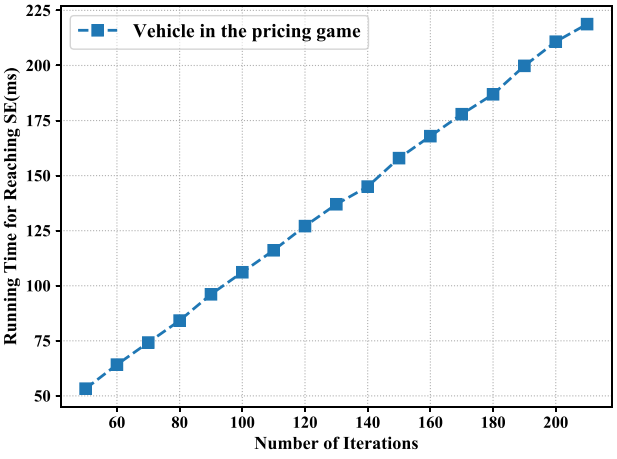

Fig. 4. Average running time with the increasing number of iterations.

To sum up, from the viewpoints of vehicles, both the amount of workloads to be undertaken and the utility values decrease with the increasing number of iterations, until the SE of the game reaches. Furthermore, the SE of the game can be sought in almost real time. After reaching the SE of the game, the utility values for these vehicles

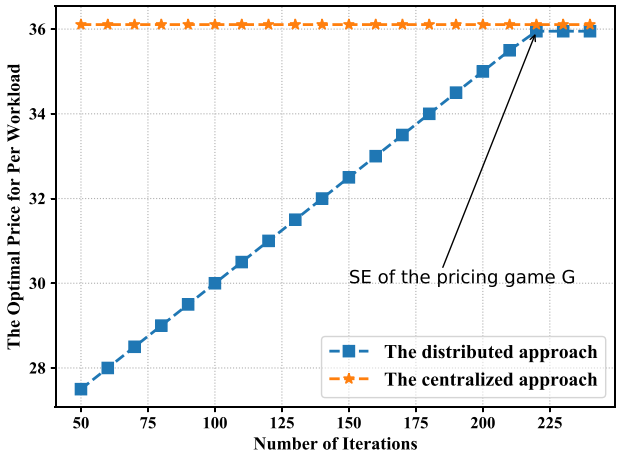

Fig. 5. Performance comparison w.r.t. price between two different approaches.

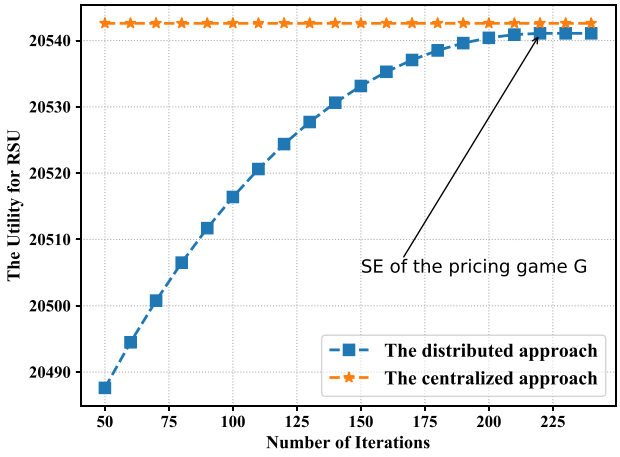

Fig. 6. Performance comparison w.r.t. utility between two different approaches.

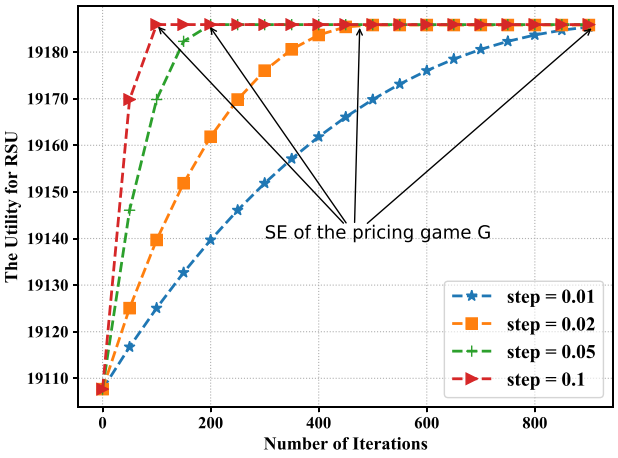

Fig. 7. Performance comparison w.r.t. step increment.

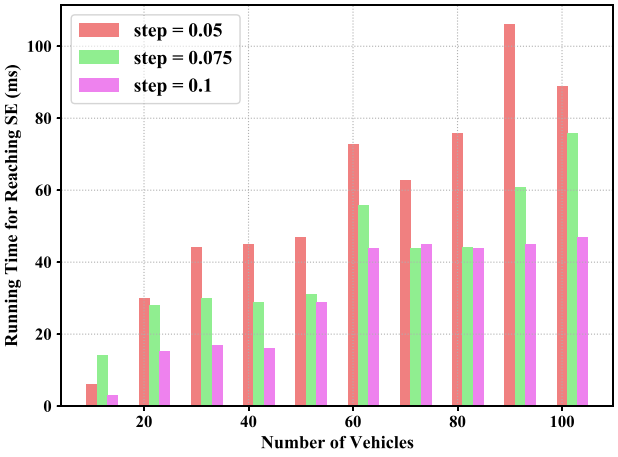

Fig. 8. Performance comparison w.r.t. the number of vehicles.

do not change anymore, which means that no vehicle can achieve a higher utility value by adjusting its own strategy unilaterally in the SE of the game.

6.2. Performance evaluation for RSU side

For the RSU side, the price per workload and the utility are also investigated. The experimental results are shown in Fig. 5 and Fig. 6, respectively. The centralized approach serves as the benchmark algorithm to evaluate the performance of the distributed algorithm for real-time resource pricing in VEC. As mentioned earlier, RSU is assumed to know the satisfaction levels of all vehicles and utility functions in the centralized approach. According to Eq. (22), the optimal price per workload is determined at the beginning as shown in Fig. 5. Similarly, the optimal utility value for RSU in the centralized approach is also determined at the beginning, as shown in Fig. 6. It is revealed from Figs. 5 and 6 that both the optimal price and the optimal utility value increase when the price per workload increases. However, when the SE of the game arrives, the optimal utility value for RSU does not increase any more as the price per workload increases. Similarly, the optimal price per workload remains the same even if the price per workload increases. For example, a mutually satisfactory result (i.e., the SE of the game) is reached in this experiment, when the number of iterations comes to 220.

To sum up, the benchmark algorithm knows the satisfaction levels of all vehicles and utility functions all the time, while the distributed approach knows neither of them for the purpose of privacy protection. Therefore, it needs to take a certain amount of time to converge to the SE of the game. However, the time is acceptable as investigated in Fig. 4. On another hand, the convergence rate of the distributed approach is also restricted by other factors, such as the way p is updated and the number of vehicles. For instance, Δp should be determined carefully, since small values of Δp slow down the convergence rate while large values of Δp could miss the convergence point. Furthermore, the number of vehicles (i.e., applications) may affect the performance of the distributed approach in terms of the convergence rate. We will investigate the effects of these parameters upon the convergence rate in the next subsection.

6.3. The effects of other parameters on pricing game

Two sets of experiments have been carried out to evaluate the effects of the step and the number of applications upon the performance of the distributed approach. To be more specific, the experimental results are shown in Fig. 7 and Fig. 8, respectively. Fig. 7 shows the influence of the way how p is updated, while Fig. 8 shows the influence of the number of applications upon the distributed approach.

In the first set of experiments, the price is increased by a step of 0.01, 0.02, 0.05 and 0.1 respectively. It is obvious that a larger value indeed helps reach the SE of the game at a higher rate. For instance, when the step is 0.1, the SE of the game can be reached with the number of iterations equal to 100. Compared to the step of 0.1, the step

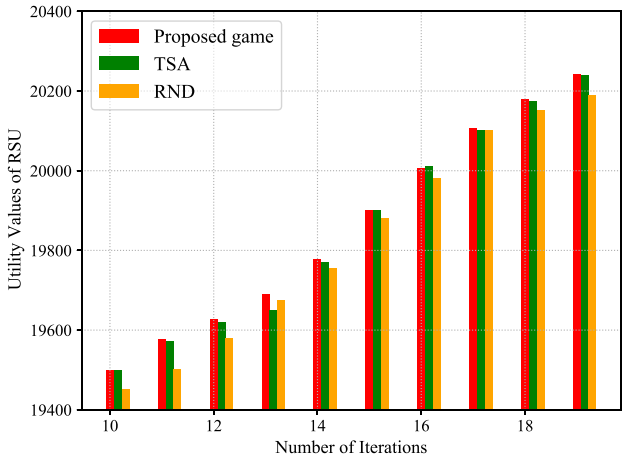

Fig. 9. Performance comparison w.r.t. utility values at RSU.

of 0.01 helps reach the SE of the game with the number of iterations equal to about 930. By contrast, the former is about nine times faster than the latter. Nevertheless, it does not mean that the larger the step, the better the performance of the distributed approach. Despite a faster convergence rate, the approach may miss the convergence point. We cannot determine the optimal value for the step, but the value can be carefully set empirically. As far as our experiment is concerned, the step equal to 0.1 is currently the best one.

In the second set of experiments, we investigate the number of vehicles (i.e., the applications) that could affect the performance of the distributed approach in terms of running time. The experimental result is shown in Fig. 8. In this experiment, the number of vehicles ranges from 10 to 100. For each application, three update strategies are applied to the distributed approach, i.e., the steps are 0.05, 0.075, and 0.1, respectively. From this figure, we can observe that (1) The average running time increases roughly when the number of applications increases, no matter which value the step is; (2) As far as the update strategies are concerned, the step of 0.1 is undoubtedly the best, no matter how many vehicles are considered; (3) The response is almost real-time even if the number of vehicles is very large. For instance, when the number of vehicles is 100, the running time is 88, 75 and 46, respectively.

To sum up, both the way the price is updated and the number of applications can affect our approach in terms of the convergence rate and the running time. Generally, the SE of the game can be reached in almost real time as anticipated. Furthermore, compared to the centralized approach, the distributed approach can efficiently solve the real-time computational resource pricing issue while considering the privacy of vehicles.

6.4. Approach comparison

We have investigated the advantage of the distributed approach over the centralized approach, as well as the effects of involved parameters upon the distributed approach. Actually, there are other approaches for seeking the SE in the proposed game. In the following, we investigate the performance of our approach such as the optimal values at the edge, compared to other approach. In particular, two approaches are used as the contrast, i.e., a ternary search based approach (TSA) [9] and a random approach (RND). TSA sets four points, i.e., the lower and upper bounds, two other values between them, so as to speed up the searching process. However, there is no upper bound of the price in our problem. To adjust TSA to our optimization, we in the simulation assume that the value of the upper bound is twice that of the lower bound, and the lower bound of the price is the same as that in DADW and DAPW. For the random approach, the increment in the price per workload is generated in a random way, while the price is increased by a step of 0.1 in DAPW. The number of vehicles is 20 in the simulation.

The simulation result on performance comparison is shown in Fig. 9, where the x-coordinate denotes the number of iterations and y-coordinate denotes the utility values of RSU. As far as the capability to find the optimal utility values is concerned, our approach is slightly better than TSA in general, while the random approach has the worst performance among the three approaches. However, due to the randomness, the random approach sometimes demonstrates better performance. For example, it is slightly better than TSA when the number of vehicles is 14. To sum up, our approach has shown better performance w.r.t. the capability to obtain the optimal values at the edge server.

7. Conclusion

The monetary reward has always been one of the most important goals for computational resource providers in VEC. The revenues can actually stimulate the providers to provide highly qualified services for the offloaded vehicular applications. In this paper, we have proposed an optimal computational resource pricing in VEC and aim to optimize the utility values of both vehicles and the edge. To achieve this goal, a Stackelberg game is applied for modeling the interactions between vehicles and the edge. Furthermore, a distributed algorithm is put forward to solve the real-time pricing game, which guarantees that the private information of vehicles can be protected. We have proven the existence of the Stackelberg equilibrium in our resource pricing game theoretically and experimentally. For the future work, we plan to design more efficient and scalable pricing schemes revolving around VEC when tasks are offloaded and computation is undertaken at the edge server.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China under Grant 61801325 and 62071327.

References

[1] C. Tang, C. Zhu, X. Wei, H. Wu, Q. Li, J.J.P.C. Rodrigues, Intelligent resource allocation for utility optimization in RSU-empowered vehicular network, IEEE Access 8 (2020) 94453–94462.

[2] P. Zhou, T. Braud, A. Zavodovski, Z. Liu, X. Chen, P. Hui, J. Kangasharju, Edge-facilitated augmented vision in vehicle-to-everything networks, IEEE Trans. Veh. Technol. 69 (2020) 12187–12201.

[3] J. Feng, Z. Liu, C. Wu, Y. Ji, Mobile edge computing for the internet of vehicles: Offloading framework and job scheduling, IEEE Veh. Technol. Mag. 14 (2019) 28–36.

[4] J. Zhang, H. Guo, J. Liu, Y. Zhang, Task offloading in vehicular edge computing networks: A load-balancing solution, IEEE Trans. Veh. Technol. 69 (2020) 2092–2104.

[5] C. Chen, L. Chen, L. Liu, S. He, X. Yuan, D. Lan, Z. Chen, Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks, IEEE Access 8 (2020) 18863–18873.

[6] Y. Dai, D. Xu, S. Maharjan, Y. Zhang, Joint load balancing and offloading in vehicular edge computing and networks, IEEE Internet Things J. 6 (2019) 4377–4387.

[7] Z. Liu, X. Zhang, J. Zhang, D. Tang, X. Tao, Learning based fluctuation-aware computation offloading for vehicular edge computing system, in: 2020 IEEE Wireless Communications and Networking Conference (WCNC), 2020, pp. 1–7.

[8] J. Zhang, X. Huang, R. Yu, Optimal task assignment with delay constraint for parked vehicle assisted edge computing: A stackelberg game approach, IEEE Commun. Lett. 24 (2020) 598–602.

[9] X. Wang, X. Chen, W. Wu, N. An, L. Wang, Cooperative application execution in mobile cloud computing: A Stackelberg game approach, IEEE Commun. Lett. 20 (2016) 946–949.
[10] C. Wu, Z. Liu, F. Liu, T. Yoshinaga, Y. Ji, J. Li, Collaborative learning of communication routes in edge-enabled multi-access vehicular environment, IEEE Trans. Cogn. Commun. Netw. 6 (2020) 1155–1165.
[11] X. Gu, G. Zhang, Energy-efficient computation offloading for vehicular edge computing networks, Comput. Commun. 166 (2021) 244–253.
[12] X. Huang, R. Yu, D. Ye, L. Shu, S. Xie, Efficient workload allocation and user-centric utility maximization for task scheduling in collaborative vehicular edge computing, IEEE Trans. Veh. Technol. 70 (2021) 3773–3787.
[13] J. Zhang, H. Guo, J. Liu, Adaptive task offloading in vehicular edge computing networks: a reinforcement learning based scheme, Mob. Netw. Appl. 25 (2020) 1736–1745.
[14] H. Guo, J. Liu, J. Ren, Y. Zhang, Intelligent task offloading in vehicular edge computing networks, IEEE Wirel. Commun. 27 (2020) 126–132.
[15] C. Sonmez, C. Tunca, A. Ozgovde, C. Ersoy, Machine learning-based workload orchestrator for vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (2021) 2239–2251.
[16] L. Zhao, K. Yang, Z. Tan, X. Li, S. Sharma, Z. Liu, A novel cost optimization strategy for SDN-enabled UAV-assisted vehicular computation offloading, IEEE Trans. Intell. Transp. Syst. 22 (2021) 3664–3674.
[17] Y. Wu, J. Wu, L. Chen, J. Yan, Y. Luo, Efficient task scheduling for servers with dynamic states in vehicular edge computing, Comput. Commun. 150 (2020) 245–253.
[18] G. Li, Y. Zhang, M. Wang, J. Wu, Q. Lin, X. Sheng, Resource management framework based on the Stackelberg game in vehicular edge computing, Complex 2020 (2020) 8936064:1–8936064:11.
[19] S. Batewela, C. Liu, M. Bennis, H.A. Suraweera, C.S. Hong, Risk-sensitive task fetching and offloading for vehicular edge computing, IEEE Commun. Lett. 24 (2020) 617–621.
[20] J. Feng, Z. Liu, C. Wu, Y. Ji, AVE: Autonomous vehicular edge computing framework with ACO-based scheduling, IEEE Trans. Veh. Technol. 66 (2017) 10660–10675.
[21] T. Cai, J. Li, A.S. Mian, R. li, T. Sellis, J.X. Yu, Target-aware holistic influence maximization in spatial social networks, IEEE Trans. Knowl. Data Eng. (2020) 1, http://dx.doi.org/10.1109/TKDE.2020.3003047.
[22] T. Wang, Y. Mei, X. Liu, J. Wang, H. Dai, Z. Wang, Edge-based auditing method for data security in resource-constrained Internet of Things, J. Syst. Archit. 114 (2021) 101971.
[23] C. Tang, C. Zhu, H. Wu, X. Wei, Q. Li, J.J. Rodrigues, A game theoretical pricing scheme for vehicles in vehicular edge computing, in: 2020 16th International Conference on Mobility, Sensing and Networking (MSN), 2020, pp. 17–22.
[24] S. Chen, L. Li, Z. Chen, S. Li, Dynamic pricing for smart mobile edge computing: A reinforcement learning approach, IEEE Wirel. Commun. Lett. 10 (2021) 700–704.
[25] D.T. Nguyen, L.B. Le, V. Bhargava, Price-based resource allocation for edge computing: A market equilibrium approach, IEEE Trans. Cloud Comput. 9 (2021) 302–317.
[26] T. Wang, Y. Lu, J. Wang, H. Dai, X. Zheng, W. Jia, EIHDP: Edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for IoT systems, IEEE Trans. Comput. 70 (2021) 1285–1298.
[27] C. Su, F. Ye, Y. Zha, T. Liu, Y. Zhang, Z. Han, Matching with contracts-based resource trading and price negotiation in multi-access edge computing, IEEE Wirel. Commun. Lett. 10 (2021) 892–896.
[28] M. Siew, D.W.H. Cai, L. Li, T.Q.S. Quek, A sharing-economy inspired pricing mechanism for multi-access edge computing, in: IEEE Global Communications Conference, GLOBECOM 2020, Virtual Event, Taiwan, December 7–11, 2020, IEEE, 2020, pp. 1–6.
[29] M. Liu, Y. Liu, Price-based distributed offloading for mobile-edge computing with computation capacity constraints, IEEE Wirel. Commun. Lett. 7 (2018) 420–423.
[30] B. Liang, R. Fan, H. Hu, Y. Zhang, N. Zhang, A. Anpalagan, Nonlinear pricing based distributed offloading in multi-user mobile edge computing, IEEE Trans. Veh. Technol. 70 (2021) 1077–1082.
[31] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM Trans. Netw. 24 (2016) 2795–2808.
[32] C. Tang, M. Hao, X. Wei, W. Chen, Energy-aware task scheduling in mobile cloud computing, Distrib. Parallel Databases 36 (2018) 529–553.
[33] T. Wang, P. Wang, S. Cai, X. Zheng, Y. Ma, W. Jia, G. Wang, Mobile edge-enabled trust evaluation for the Internet of Things, Inf. Fusion 75 (2021) 90–100.
[34] C. Tang, C. Zhu, H. Wu, Q. Li, J.J. Rodrigues, Towards response time minimization considering energy consumption in caching assisted vehicular edge computing, IEEE Internet Things J. (2021) 1, http://dx.doi.org/10.1109/JIOT.2021.3108902.

Chaogang Tang received his B.S. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, and Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China, Hefei, China, and the Department of Computer Science, City University of Hong Kong, under a joint Ph.D. Program, in 2012. He is now with the China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, Internet of Things, big data.

Huaming Wu received the B.E. and M.S. degrees from Harbin Institute of Technology, China in 2009 and 2011, respectively, both in electrical engineering. He received the Ph.D. degree with the highest honor in computer science at Freie Universität Berlin, Germany in 2015. He is currently an associate professor in the Center for Applied Mathematics, Tianjin University. His research interests include model-based evaluation, wireless and mobile network systems, mobile cloud computing and deep learning.