Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

Resource pooling in vehicular fog computing

Chaogang Tang[1], Shixiong Xia[1*], Qing Li[2], Wei Chen[1] and Weidong Fang[3]

Abstract
Vehicular fog computing (VFC) provisions computing services at the edge of networks by fully exploiting the idle resources of vehicle loaded computer systems. Task scheduling and resource allocation revolved around VFC have gained tremendous attention recently. Currently, most of these works in VFC have focused on response time optimization or energy reduction. Computing services are provisioned in a pay-as-you-go model and vehicles as resource contributors are stimulated by the benefits obtained by leasing these resources. How to maximize their own benefits is one of big concerns but few of current works have recognized its importance in VFC. We in this paper introduce the notion of resource pooling into VFC where the computing resources of vehicles are pooled together to jointly provision computational services in a community. A genetic algorithm based strategy is proposed to solve the optimization problem for the sake of benefit maximization. Extensive experiments have been carried out to evaluate the approach and the numeric results have demonstrated that our strategy outstands other approaches with regards to the optimization objective.

Keywords: Vehicular fog computing, Resource pooling, Provision, Task scheduling, Resource allocation

Introduction

The Internet of Things (IoTs) defines a connection paradigm, where people and things are able to connect and communicate anytime, anyplace with anything and anyone, ideally using any network and any services [1, 2]. The development of information and communication technology (ICT) and IoT give rise to great revolution of the way of life for urban residents, e.g., humanized services are available and living standards are further improved. With the development of IoTs, the concept of smart city is then proposed to cater for the demanding of making an instrumented, interconnected, and intelligent city. Against this background, a vast amount of data is generated by IoT devices which needs real-time analysis and processing. Although the remote cloud center is efficient at mining and extracting valuable knowledge from these data, task execution in cloud usually brings about long response latency, owing to the transmission delay caused by data transmission via the backhual links. Task processing in this way is not appropriate for explosively increasing applications featured by strict latency requirement. As a result, new computing paradigms are urgently needed to support such applications which are generated either from IoT devices or mobile terminals [3].

In this context, vehicular fog computing (VFC) [4–6] is proposed to provision computing services at the logical edge of networks, e.g., by fully exploiting the idle resources of vehicle loaded computer systems. The rationale behind VFC is that vehicles have evolved to the point where they are capable of data processing, analysis, and reasoning with the aid of powerful computing and communication facilities. Specifically, VFC usually consists of mobile vehicles and immobile infrastructures such as road side unit (RSU). Usually RSU has more powerful computing capabilities than vehicles. Task scheduling and resource allocation revolved around VFC have gained tremendous attention in the past few years [7–12].

*Correspondence: xiasx@cumt.edu.cn
[1]School of Computer Science and Technology, China University of Mining and Technology, Daxue Road, 221000 Xuzhou, China
Full list of author information is available at the end of the article

Most of these works focus on energy-aware and time-saving task offloading and resource allocation in VFC. On one hand, data generated by IoT devices can be offloaded and analyzed in VFC for the sake of response latency optimization; on the other hand, owing to the inherent defects of smart mobile terminals such as limited computational resources and constrained battery energy, tasks from these smart terminals (e.g., smart watches, smart phones, and PDAs) can also be outsourced to VFC for execution. In terms of the second case, VFC is similar to the mobile cloud computing (MCC) [13–15] that offloads tasks hosted at these terminals to the cloud for execution in hope to mitigate the energy consumption of mobile devices. However, compared to MCC, VFC has much shorter response latency owing to the distribution of computing resources at the edge.

Although task offloading and resource allocation in VFC have been studied extensively, we have observed that few of these works have investigated this problem from the perspective of resource providers (e.g., vehicles). In VFC, computing services are provisioned in a pay-as-you-go model, which means consumers need to pay for the requested resources. Accordingly, as resource contributors, vehicles in VFC are always stimulated by these considerable benefits. How to maximize their own benefits is one of big concerns when vehicles provision computing services.

On another hand, in VFC, large numbers of resource requests are sent to the fog nodes, which needs frequent interactions between requestors and fog vehicles. And it may incur tremendous stress over the communication resources. In addition, it is the responsibility of requesters for deciding which fog vehicle to offload the tasks, and the offloading decision may not be globally optimal due to lack of global information upon the fog vehicles in VFC.

To tackle these issues, we in this paper introduce the notion of *resource pooling* into VFC where the computing resources of vehicles are pooled together to jointly provision computational services in a community [16]. A community can be sponsored by an immobile infrastructure (e.g., RSU). Generally speaking, RSU can serve as a fog server with multiple functionalities. For instance, it can recruit nearby vehicles to join the community by broadcasting the beacon information. The beacon information includes the information about its vehicle members, the available resources, and the benefits for contributing the resources. Vehicles can join or leave the community for free. Second, RSU holds the global information about its members and separates the control flow and data flow, following the principle of service-oriented architecture (SOA). To be specific, RSU receives and processes the resource requests, and makes decisions on task offloading for these requests. Then, tasks are offloaded to the designated fog vehicles directly. Thus, each entity in the community is dedicated to its own business.

Resource pooling in VFC can boost the efficiency, flexibility and reliability of the VFC systems. The deployment of RSUs especially in the densely populated area increases the probability that one vehicle can be covered by multiple RSUs. In such a situation, from the perspective of fog vehicles, how to select a community to join is really a big concern to them, as vehicles expect more benefits by joining the community. Considering the high dynamics of network topology, limited wireless communication range, and different pricing in different communities, the decision making on community selection is not trivial.

The rest of paper is organized as follows. We have studied the related works in "Related works" section. Some preliminaries are given in "Preliminaries" section. In "System model and problem formlation" section, system model is introduced and the optimization problem is mathematically formulated. A genetic algorithm based strategy on decision making is put forward in "Proposed solution" section and experimental evaluation is reported in "Numeric evaluation and results analysis" section. Finally, the conclusion comes in "Conclusion" section.

Related works

Fog computing constitutes one of the key enablers for smart cities, by endowing edge devices with certain computing and storing abilities. VFC is derived from fog computing where the edge devices are the mobile vehicles. Considering the mobility of vehicles, task offloading and resource allocation are characterized by limited service time in VFC. As a result, task offloading and execution in vehicle fogs are important and complicated, which has attracted extensive attention recently.

In the previous works [5, 7, 16], VFC has been investigated from multiple viewpoints, e.g., latency optimization, utility maximization and so on. To be specific, intelligent transportation system has been developing rapidly recently, with the purpose of providing drivers with a convenient, economical and environmentally friendly environment. Explosive growth in the number of vehicular applications causes tremendous stress over the limited computing capabilities of vehicles. To cope with computationally intensive and time sensitive in-car applications, a resource allocation algorithm is put forward in order to optimize the utility [7].

In [16], an application scenario in VFC is considered, in which vehicles try to get their own benefits by contributing their idle computing resources. RSU is responsible for decision making for task scheduling. Meanwhile, an algorithm is proposed following the principle of service oriented architecture. The numeric results have revealed enormous advantage in comparison with other strategies.

Furthermore, edge services (e.g., RSUs in the context of VFC) and cloud center can also cooperatively provision computing services to the requestors such as vehicles in vicinity for multiple purposes. In [5], vehicular requests will be served at RSUs at the beginning and they are forwarded to the cloud center if the service of quality (QoS) at RSU is no longer qualified. For example, explosive number of vehicular requests sometimes causes RSU resource shortage in short time.

VFC is characterized by the high mobility of vehicles in road, so vehicles sometimes may leave the communication range before the offloaded tasks are accomplished. It will incur the inefficiency and low success rate of VFC and yet few of works have considered it in existing works. To tackle this problem, authors in [17] investigate the offloading scheme in VFC system with consideration of the departure of vehicles with offloaded tasks. They formulate it as a semi-Markov decision process that is solved by a designed value iteration algorithm targeted at optimizing the total reward of the system.

Owing to the limited computing capabilities of vehicles in VFC, on one hand, moving vehicles are busy with service provisioning; on the other hand, they also are supposed to provide real-time responses for traffic and accident warnings in a real-time fashion. Against this background, computing resources may be in short supply sometimes. Therefore, authors in [18] propose to utilize the computing resources of parked vehicles to supplement VFC. A heuristic algorithm combined with reinforcement learning is put forward to solve the formulated problem, e.g., by learning the vehicles' movement and parking status in smart city. Similar works on resource allocation can also be found in [19, 20].

Shortcomings in VFC such as limited communication bandwidth, high mobility, huge data transmission, and overwhelming computation overhead make it difficult for vehicles to deliver the services according to the declared QoS [21]. Thus, authors in [21] propose a VFC architecture to exploring possibility of working together among the cloud, static fog and mobile fog. They introduce the system model in this architecture to quantitatively analyze its characteristics. On another hand, considering the information of uncertainty, to make task offloading reliable is still an important yet challenging issue. Authors in [22] propose a task offloading algorithm based on matching learning to tackle this problem. They try to optimize the response latency using the pricing-based matching, and design a matching-learning-based algorithm for task offloading in VFC.

Several application scenarios can also be found in [23–25] where vehicles as fog nodes realize multiple functionalities such as real-time traffic management, congestion detection and so on. For instance, massive efforts are dedicated to the congestion handling on the road. Leveraging connected vehicles for congestion control has proved to be practically feasible. To be specific, congestion detection and handing with an aid of connected vehicles technology is investigated in [25], in order to detect congestion while satisfying traffic management requirements.

Different from the aforementioned works, we in this paper consider VFC with a pay-as-you-go feature and strive to stimulate resource contributors (i.e., vehicle) by maximizing their own benefits. Specifically, to realize resource pooling at community, efficient algorithm is designed for deciding which community for vehicles to join.

Preliminaries

We motivate our work by an example in the following. Suppose that there are four communities as shown in Fig. 1, where four RSUs have sponsored their own communities to attract the nearby vehicles. Several vehicles under the coverage of communities want to join the

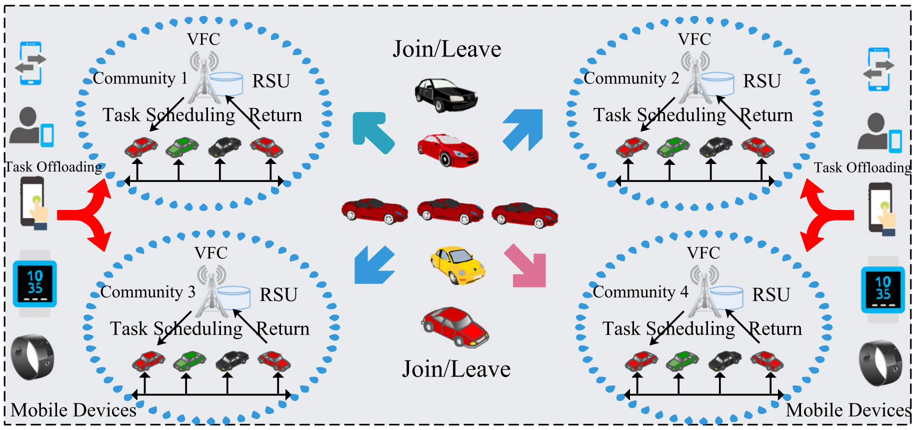

Fig. 1 Architecture of task scheduling in community

community such that they can obtain benefits by contributing the computing resources in the communities. Then a question is posed naturally concerning which community can deliver the greatest benefits for the vehicles. Several factors which affect the benefits of vehicles can be outlined as follows:

- *Dwell time.* It denotes the time vehicles can stay within the communication range of community. It is an important metric to evaluate the service time of vehicles. Intuitively, the longer the service time, the more the benefits.
- *Available resources.* Vehicles can obtain benefits by contributing their idle resources. It is one of the most important metrics which affect the benefits of vehicles. The more the amount of available resources they contribute, the more the benefits they can earn.
- *Unit price.* The pricing for unit resource in different communities may be different. Vehicles tend to join the community with higher pricing. Higher pricing brings about more benefits for vehicles.
- *Strategy of competitors.* Given a vehicle v, its benefits in the community can be affected by other vehicles in the same community. The task scheduling and resource allocation in the community follows the principle of SOA and RSU is responsible for decision making on which community member (i.e., fog vehicle) is designated to respond to the request and perform the corresponding task. If the number of fog vehicles in the community becomes large, the probability that v is chosen to perform the task is also decreased. As a consequence, the decisions of other fog vehicles on whether or not to join the community will affect the benefits of v.
- *Number of task requests.* The arrival rate of task requests at each community may be different. Generally speaking, high arrival rate brings about high frequency of task allocation, which increases the benefits of fog vehicles within the same community to some extent.

Before joining the community, vehicles should take into consideration these factors comprehensively. Due to the wide deployment of RSUs, the number of communities which vehicles can join at the same time is explosively increasing. It is not an easy work for vehicles to select the suitable community to contribute the computing services, especially considering the potential benefits. For instance, as shown in Fig. 1, if the number of vehicle members in Community 1 is much more than other communities, it is unadvisable for the vehicles on the road to join Community 1. On anther hand, if most of the vehicles on road choose to join Community 1, it is also unadvisable for vehicle, say v, to join Community 1 at the same time.

Vehicle-to-Vehicle (V2V) and Vehicle-to-RSU (V2R) communication technologies enable information sharing and data delivery among these entities. RSU usually has all the information about its members in the community while vehicles can learn the information about the community during the beacon information exchange. To be specific, the procedure of interaction between RSU and vehicles can be detailed as follows. At the beginning, RSU broadcasts the beacon information $bcnMsg = (cmt\_ID, cmt\_loc, no\_mbr, avl\_rsc, p, timestamp)$ to the nearby vehicles. cmt_ID and cmt_loc represents the numeric identification and location of the community, respectively. no_mbr denotes the current number of members in the community. avl_rsc is the computing resources available in the community. p is price that resource requesters need to pay for using unit computing resource. Note that p may be a function to better reflect the real-world pricing mechanism.

From the descriptions above, we can observe that to realize the benefits maximization from the perspective of vehicles, the information obtained from communities is not enough to aid the best decision making. For example, recall the factors affecting the benefits of vehicles, and the benefits of a vehicle can be influenced by other vehicles' decisions. However, the information pertaining to other vehicles' decisions is not included in *bcnMsg*. In such cases, vehicles cannot make decisions immediately after they receive the beacons, for the reason that they need to reason and evaluate the communities combining other information such as the decisions of other vehicles. To this end, we assume that vehicles under the communication coverage of the same RSU can share the decisions freely and honestly by V2V communication technologies. Each vehicle learns from these information and makes the best decisions for themselves.

After the decision is made, confirmation information can be sent to the corresponding community, and finally the vehicle becomes a member of the community. In this paper, our work focuses on the phase of decision making from the perspective of vehicles. As smart agents, vehicles need to make the best decisions for themselves.

System model and problem formlation

There is a set of RSUs $R = \{R_1, R_2, \cdots, R_m\}$ and a set of nearby vehicles $V = \{v_1, v_2, \cdots, v_n\}$, where m and n denote the number of RSUs and vehicles, respectively. Each RSU R_i can initialize a community C_i for the sake of computing resource pooling. To recruit the vehicles, beacons *bcnMsg* are broadcast by communities using V2R technologies such as DSRC, 4G/5G, Zigbee, Bluetooth, etc. Each vehicle can join at most one community at the same time. Assume that vehicles are equipped with GPS receivers such that the location information can be known and disseminated at any time. After receiving *bcnMsg* from the communities, vehicles start to evaluate the communities with an aim to pursue the benefits maximization. The decision should be made based on the information gathered from all the involved entities (e.g., RSU, vehicles). For example, the information dissemination and sharing is needed, so vehicles can learn the decisions of other vehicles.

Due to the high mobility of vehicles, the time during which a vehicle can stay in one community is limited. Given a vehicle v_i, denoted by $dt_{i,j}$ the dwell time of v_i in the community C_j and $dt_{i,j}$ can be calculated as follows:

$$dt_{i,j} = \frac{2D_j \sin(\theta_{i,j}/2)}{\rho_i} \tag{1}$$

where D_j is the communication distance of Community C_j, ρ_i is the velocity of v_i, and $\theta_{i,j}$ as shown in Fig. 2, is the angle formed between R_j and the points where v_i enters and leaves the communication range of C_j, respectively. $\theta_{i,j}$ can be obtained in advance, since it is relatively fixed. As observed in Fig. 2, $\theta_{i,j}$ is dependent upon the location of RSU relative to the road rather than the vehicles.

However, if the distance between v_i and C_j, denoted by $dis(i,j)$, is larger than D_j, it is unlikely for v_i to join C_j. Thus, the dwell time $dt_{i,j}$ can be generally defined as:

$$dt_{i,j} = \begin{cases} \frac{2D_j \sin(\theta_{i,j}/2)}{\rho_i} & dis(i,j) \le D_j \\ 0 & dis(i,j) > D_j \end{cases} \tag{2}$$

If vehicle v_i joins the community C_j, its computing resources will be contributed to C_j. In this way, C_j can pool resource together and thus provision computing services to the nearby requestors. Given a unit time, the benefits of v_i in C_j, denoted by $B_{i,j}$, can be defined as follows:

$$B_{i,j} = \lambda_j \cdot p_j \cdot \frac{r_i}{\sum_{v_k \in C_j} r_k} \tag{3}$$

where λ_j denotes the average amount of computing resources at C_j required by requesters and it can be set in advance according to the historical experience, p_j is the price that requesters need to pay for using unit computing resource of C_j. r_i represents the amount of computing resources that v_i can contribute to C_j. In this paper, to simplify the discussion, we assume that the benefits are distributed among the community members proportionally to the corresponding resources they can contribute. Based on the definition, we can see that the benefits of v_i can be affected by the strategies of other vehicles in C_j. The more the number of vehicle members in C_j, the less the benefits v_i can earn.

As mentioned earlier, several factors should be considered such as dwell time, available resources and so on, when vehicles pursue the maximal benefits. Accordingly, we define the utility of vehicle v_i in the community C_j as:

$$\mathcal{U}_{i,j} = dt_{i,j} \cdot B_{i,j} - r_i \cdot c_i \tag{4}$$

where c_i denotes the cost for contributing unit computing resource to C_j.

The objective of vehicles is to maximize their benefits by joining the suitable communities. A decision variable $\phi_i = (\phi_{i,1}, \phi_{i,2}, ..., \phi_{i,m})$ is used to denote the decision of vehicle v_i, where $\phi_{i,k}$ $(1 \le k \le m)$ is binary variable. If v_i chooses to join community C_k, $\phi_{i,k} = 1$; otherwise, $\phi_{i,k} = 0$. Given a decision profile $\boldsymbol{\phi} = (\phi_1, \phi_2, ..., \phi_n)$ of all vehicles in V, the utility of the vehicles in V can be mathematically formulated as follows:

$$\mathcal{U}(\boldsymbol{\phi}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \phi_{i,j} \cdot \mathcal{U}_{i,j} \tag{5}$$

Then the optimization problem in this paper can be formulated as:

$$P: Maximize \quad \mathcal{U}(\boldsymbol{\phi}) \tag{6}$$

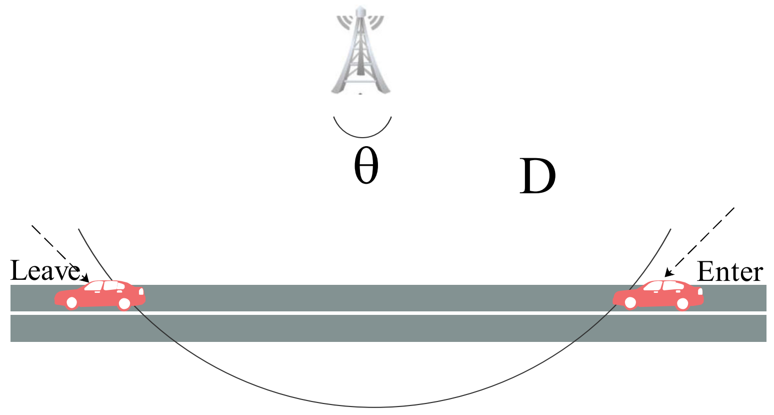

Fig. 2 An example of the dwell time for vehicle in one community

subject to:

$$\sum_{j=1}^{m} \phi_{i,j} = 1 \qquad \forall i \in [1, n] \tag{7}$$

$$\phi_{i,j} \cdot dis(i,j) \le D_j \qquad \forall i \in [1, n] \tag{8}$$

$$\mathcal{U}_{i,j} \ge 0 \qquad \forall i \in [1, n], \forall m \in [1, m] \tag{9}$$

$$\phi_{i,j} \in \{0, 1\} \qquad \forall i \in [1, n], \forall m \in [1, m] \tag{10}$$

Constraint (7) guarantees that for arbitrary vehicle in V, it can join one and only one community. If vehicle v_i wants to join C_j, it must be under the communication coverage of C_j. The locations of both v_i and C_j can be known to each other in advance, and we use $dis(i,j)$ to represent the physical distance between them. Thus, the distance should be shorter than the communication range of C_j (i.e., D_j), which can be guaranteed by constraint (8). Constraint (9) ensures that there must be the profits earned by contributing computing resources for an arbitrary vehicle in V. This constraint condition is indispensable to stimulate vehicles to join the communities.

Problem Analysis. Obviously, the potential solution space is of m^n, where m and n denote the number of RSUs and vehicles respectively. As a result, it is not advisable to seek the best solution with exhaustive algorithms. In the next section, we propose a genetic algorithm to cope with this resource pooling problem in the vehicular fog computing.

Proposed solution

In this paper, we adopt a genetic algorithm (GA) based approach to solve the problem P. GA is well known owing to its advantages in solution searching over huge potential solution space, e.g., easy deployment, powerful search capability and so on. GA is a population based iterative algorithm that mainly consists of selection, crossover and mutation operations.

As a phenotype, each individual in the population can represent a potential solution to the problem P. We encode the corresponding genotype (i.e., chromosome) in the next. In the previous problem statement, arbitrary vehicle $v_i (1 \le i \le n)$ can join arbitrary community $C_j (1 \le j \le m)$ for the sake of profit maximization. The decision of v_i is defined as ϕ_i. Thus the decision for all the vehicles is $(\phi_1, \phi_2, ..., \phi_n)$ that is actually a matrix. Accordingly, the chromosome chm can be represented by $\boldsymbol{\phi}$, i.e.,

$$chm = \begin{vmatrix} \phi_{1,1} & \cdots & \cdots & \cdots & \cdots & \phi_{1,m} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \cdots & \cdots & \phi_{2,m} \\ \phi_{3,1} & \cdots & \phi_{3,3} & \cdots & \cdots & \phi_{3,m} \\ \phi_{4,1} & \cdots & \cdots & \phi_{4,4} & \cdots & \phi_{4,m} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \phi_{n,1} & \cdots & \cdots & \cdots & \cdots & \phi_{n,m} \end{vmatrix} \tag{11}$$

where each row known as the gene segment represents the choice of an individual vehicle and thus the decision profile chm (i.e., the chromosome) can represent the choices of all the vehicles. Large numbers of chromosomes constitute the population of GA and the initial population can be generated in a random way.

Fitness function

Fitness function can quantitatively evaluate the environmental adaptability of individuals in the population in the current generation, so based on the fitness values of individuals, GA can determine which individuals should be reserved and which ones should not in the next generation.

For our problem P, each individual in the population represents a potential decision profile for all the vehicles. Thus, we can directly use Equ. (5) to act as the fitness function. Specifically, the larger the value of the fitness function, the better the environmental adaptability for the individual.

Crossover and mutation operations

Two common methods used to produce the next generation of population in GA are the crossover and mutation operations, respectively. Two parents can cooperatively produce the offspring in the next generation by the crossover operation. To be more specific, the crossover operation is to exchange the gene segments of the two parents. In this paper, a multi-point crossover operation is adopted, and an example corresponding to our problem in this paper is shown in Fig. (3). Since a chromosome is encoded into a matrix in this paper, a random binary matrix S of the same size as the chromosome is used to assist the crossover operation. Given S, check each gene position in it. If the value for this gene position is nonzero, then the two parent chromosomes exchange their gene values with each other (see Fig. 3).

On another hand, a mutation operation usually follows after a crossover operation, with an aim to preserve the diversity of individuals in the population. The mutation operation strives to generate a new individual by altering a gene randomly in the chromosome. As for our problem, we still adopt a single point mutation which is different from the previous crossover operation. The crossover

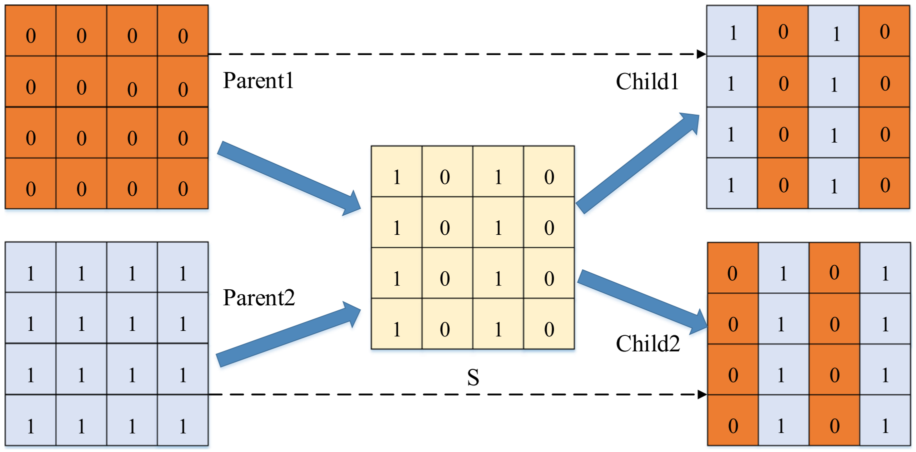

Fig. 3 An example of a multi-point crossover operation

operation strives to enhance the global search capability of GA while the mutation operation is targeted at improving the local search capability. To avoid being trapped into local optimization, we thus adopt a single point mutation operation in this paper. To that end, for a given a binary matrix S which is generated randomly, there is only one nonzero value in S, and the value for the current gene position is altered if the value for the same position in S is nonzero.

GA based decision making algorithm

In this section, we present a GA based decision making algorithm (GADM) for the problem P and the corresponding pseudo code is shown in Algorithm 1. In this algorithm, *cp* and *cm* denote the crossover probability and mutation probability, respectively. Note that in GADM the crossover and mutation operators are independent of each other, since we perform both operators over the common population S_1 (lines 7–16).

It is likely that a resulting individual is invalid after the crossover and mutation operators, due to the constraints violation (e.g., Eqs. (7)-(9)). To guarantee that each individual in the population is valid, we should check whether it violates the constraint conditions or not after each crossover and mutation operations. If the violation happens, the individual is then abandoned, and new individual should be supplemented such that the size of population is fixed.

Numeric evaluation and results analysis

Experimental settings

Extensive experiments have been carried out to evaluate the efficiency and effectiveness of the approach in this paper. The experimental design and numerical results are reported in this section, respectively. A simulator is developed and the corresponding experiments are run on a laptop with 1.8GHz Intel I5 Quad-Core CPU, 8G of RAM, Microsoft Windows 10 Operating System, Python 3.7. Some key parameter settings are listed in Table 1.

In the initial parameter settings, the number of communities is a constant (e.g., 10), for the reason that an assumption of too many overlapping communities costs too much and also is impractical in reality. In addition, to simplify the experimental design, we overlook the calculation of dwell time of each vehicle at each community, by just assuming that the dwell time of each vehicles randomly varies from 3 to 10. The population size is set to 100 and the number of iterations is 50 at the beginning.

Comparison benchmark

On one hand, as far as GADM itself is considered, several parameters can affect the performance of GADM to a great extend with regards to convergence rate, running time, optimal solution approximation and so on. For instance, these parameters usually include the crossover probability, mutation probability, population size, and the number of iterations. On the other hand, GADM should be compared with other approaches to investigate its efficiency and effectiveness. Specifically, we in this paper have adopted two approaches as the comparison benchmarks. One is the greedy approach and the other is the random approach.

For the former, there are usually several rules to guide the greedy search for the approximate optimal solution. For instance, one obvious greedy rule is that vehicles tend to join the community with longer dwell time. Intuitively, longer dwell time means longer service time, which can bring about more benefits for vehicles. Another greedy rule is that vehicles tend to join the community with larger values of λ and p, for the reason that according to the Eq. (3), larger values of λ and p can bring forth more benefits. However, it is worth mentioning that this greedy rule

Algorithm 1: GA based decision making algorithm (GADM)

```
    Input: GA involved parameters and problem $P$ involved parameters, respectively
    Output: The optimal fitness value $\mathcal{V}$
 1  Initialize a population $S$ for GA;
 2  Calculate the fitness value of each individual in the current population;
 3  Record the best fitness value $\mathcal{V}$;
 4  while Termination condition not satisfied do
 5      Perform the selection operation on $S$;
 6      Produce new population $S_1$;
 7      for each pair of individuals in $S_1$ do
 8          if $\rho > cp$ then
 9              Perform the crossover operations on this pair;
10          end
11      end
12      for each individual in $S_1$ do
13          if $\mu > cm$ then
14              Perform the mutation operation on this individual;
15          end
16      end
17      Construct the new population $S_2$ with resulting individuals;
18      Supplement $S_2$ with new individuals to keep the population size fixed;
19      Calculate the fitness values of the current population;
20      Update the best fitness value $\mathcal{V}$ if necessary;
21  end
22  Return the largest fitness value $\mathcal{V}$;
```

Table 1 Experimental parameter settings

Parameter	Description	Default value
cp	Crossover probability	0.2
cm	Mutation probability	0.01
size	Population size	100
step	The number of iterations	50
λ	Average resources to be requested	[10, 15]
n	The number of communities	10
p	The price for using unit resource	[20, 30]
r_i	The amount of resources vehicle *i* can contribute	
c_i	The cost for contributing unit resource for vehicle *i*	(0, 1)

will incur the backlog of vehicles in one community. Vehicles following this rule always join the same community at the time, which however makes the part of Eq. (3) (i.e., $\frac{r_i}{\sum_{v_k \in C_j} r_k}$) much smaller. As a result, this greedy rule is not satisfactory compared with the rule with longer dwell time. Therefore, for the greedy benchmark, we adopt the greedy rule with longer dwell time to guide the solution searching. We denote this approach by "Greedy" in this paper.

For the latter, a random approach is adopted as the random benchmark. To be specific, each vehicle selects a community to join randomly. In this way, constraint violation occurs all the time. To avoid the constraint violation, before joining the community, each constraint condition (e.g., (2), (8) and (9)) should be checked. We denote this approach by "Random" in this paper.

Numeric results and analysis

A series of experiments have been conducted following the aforementioned principles. The corresponding results are reported as follows.

Impact of parameters

The first set of experiments is investigate the impact of the crossover probability (i.e., *cp*) in this paper, and the results in terms of the fitness values and response time have been shown in Figs. 4 and 5, respectively. Given other parameters with the default values set in Table 1, we vary the crossover probability from 0.1 to 0.5 with a step of 0.1 in the experiment. From Fig. 4 we can observe that GADM can reach the best fitness value when *cp* is equal to 0.3 with the number of iterations around 20. Although other cases can obtain relatively fast convergence rate, they are easily trapped into local optimization.

For instance, when *cp* equals 0.1, GADM reaches its local optima with the number of iterations equal to 11. After numbers of iterations, GADM continues to reach a better value, but it is not as good as the case with *cp* equal to 0.3. On the other hand, the corresponding running time for each iteration is recorded and shown in Fig. 5. From the figure we can see that different values of *cp* seem to have less impact on the response time compared to the impact on the fitness values. However, when *cp* is equal to 0.5, the fluctuation of response time over each iteration is much greater than other cases. Accordingly, we can carefully draw the conclusion that *cp* with a larger value is not always better. In our experiment, GADM can reach the best performance when *cp* is 0.3, considering the fitness values and the response time.

The second set of experiments is to investigate the impact of mutation probability (i.e., *cm*) in this paper. The experimental results in terms of the fitness values and response time have been shown in Figs. 6 and 7, respectively. Similarly, given other parameters with the

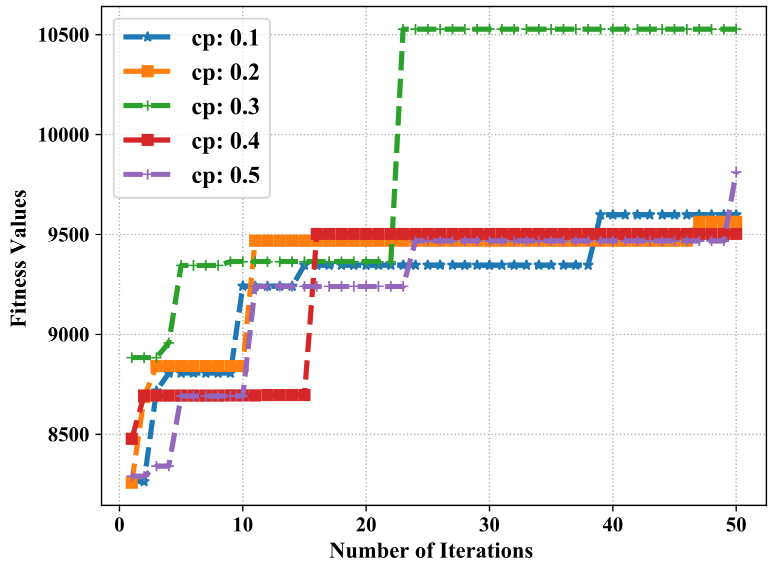

Fig. 4 Fitness values with different values of crossover probability

default values set in Table 1, we vary the mutation probability *cm* from 0.01 to 0.05 with a step of 0.01 in the experiment. As mentioned earlier, the crossover operation strives to enhance the global search capability of GA while the mutation operation is targeted at improving the local search capability. As a result, the mutation probability is usually smaller than 0.1. From Fig. 6 we can observe that when $cm = 0.05$, GADM can obtain the best fitness value compared to other cases. Moreover, fast rate of convergence is not always better, for the reason that GADM tends to fall into the local optima. Take the case with $cm = 0.04$ for example, GADM reaches its best fitness value when the number of iterations is about 11. The fitness value does not change with the increasing number of iterations.

On the other hand, the corresponding running time for each iteration is recorded and shown in Fig. 7. From the figure we can see that different values of *cm* have less impact on the response time compared to the impact on the fitness values. Accordingly, we can carefully draw the conclusion that in our experimental settings, GADM can achieve its best performance when *cm* equals 0.05.

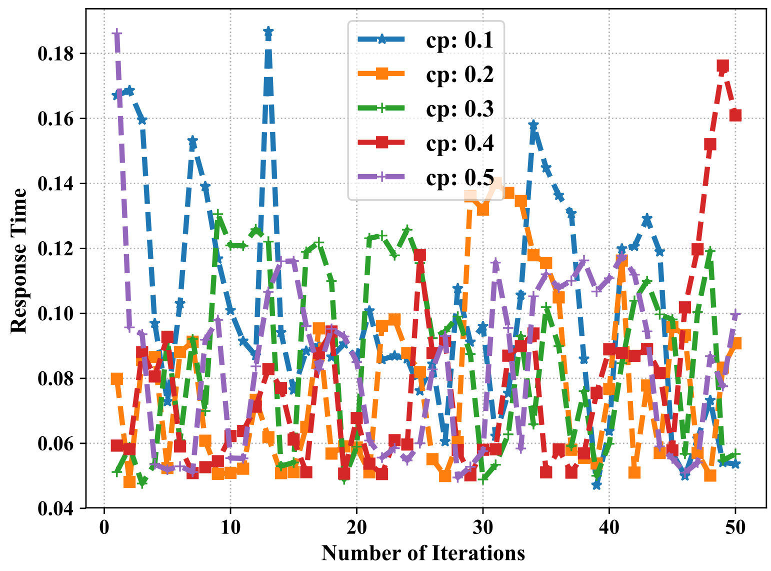

Fig. 5 Response times with different values of crossover probability

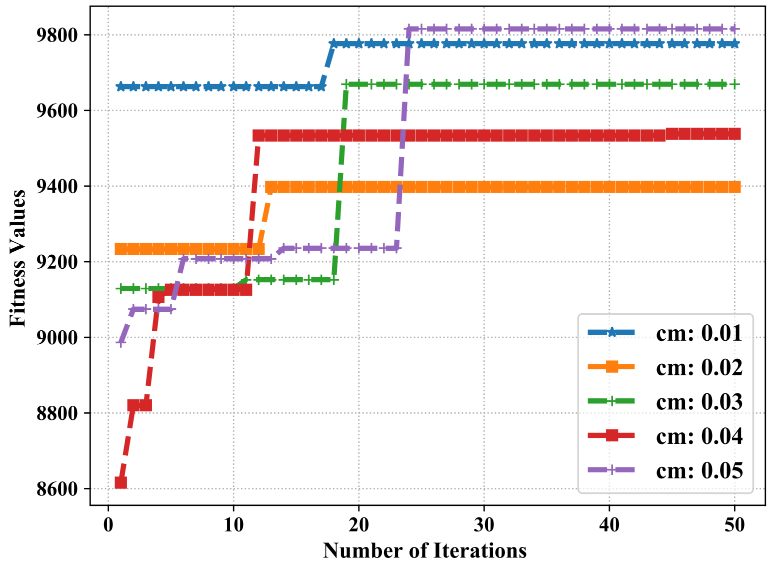

Fig. 6 Fitness values with different values of mutation probability

The third set of experiments is to investigate the impact of population size (i.e., *size*) in this paper. The experimental results in terms of the fitness values and response time have been shown in Figs. 8 and 9, respectively. Similarly, given other parameters with the default values denoted in Table 1, we vary the population size *size* from 50 to 150 with a step of 25 in the experiment. From Fig. 8 we can observe that the performance of GADM depends on the population size to a great extent. Generally, given the number of iterations, the larger the population size, the better the fitness values. For example, when the population size equals 150, the fitness value is the largest one among these cases.

Interestingly, the case with population size of 75 is worse than the case with population size of 50 in terms of the fitness values. However, this situation can be reasonable and acceptable for the following reason. The population is generated in a random way in the simulation, and thus there are no common data among these cases with different population size, which is different from the first two sets of experiments. During the investigation of impacts of *cp* and *cm* on GADM respectively, each set of

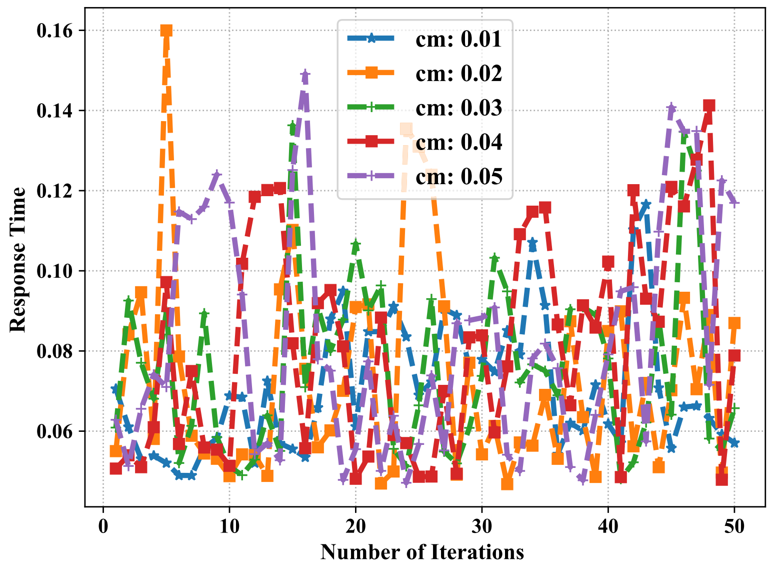

Fig. 7 Response times with different values of mutation probability

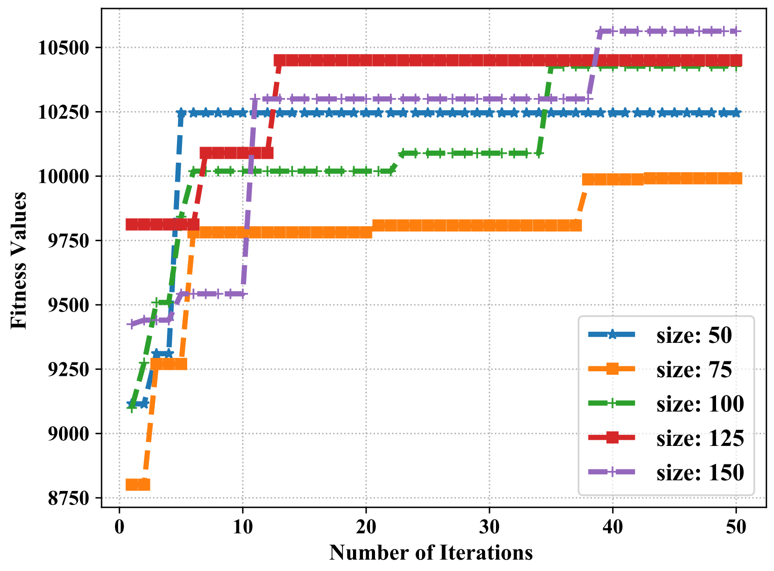

Fig. 8 Fitness values with different values of population size

experiments has the common data, i.e., the population. In this set of experiments, due to the difference in population initialization in each case, better individuals may be included in the population with small size with regards to the fitness value. As a result, an abnormal situation Interestingly, the case with population size of 75 is worse than the case with population size of 50 in terms of the fitness values. However, this situation can be reasonable and acceptable for the following reason. The population is generated in a random way in the simulation, and thus there are no common data among these cases with different population size, which is different from the first two sets of experiments. During the investigation of impacts of *cp* and *cm* on GADM respectively, each set of experiments has the common data, i.e., the population. In this set of experiments, due to the difference in population initialization in each case, better individuals may be included in the population with small size with regards to the fitness value. As a result, the abnormal situation at the beginning occasionally happens.

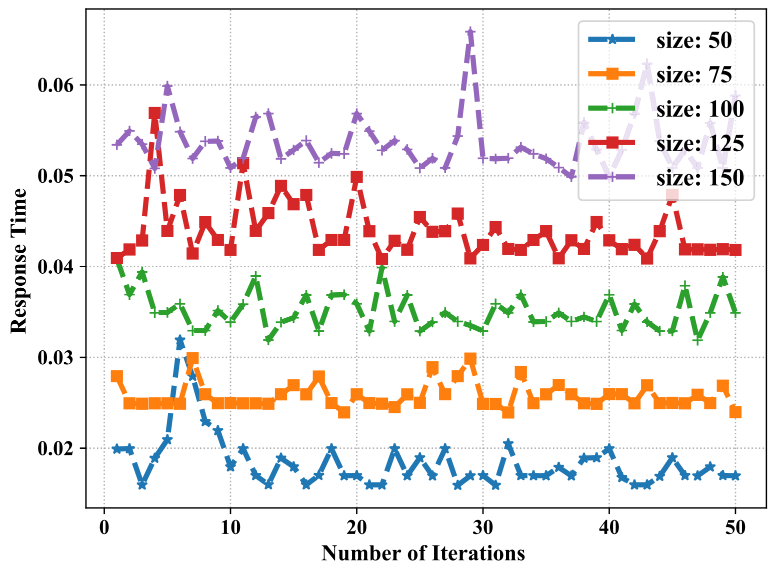

Fig. 9 Response times with different values of population size

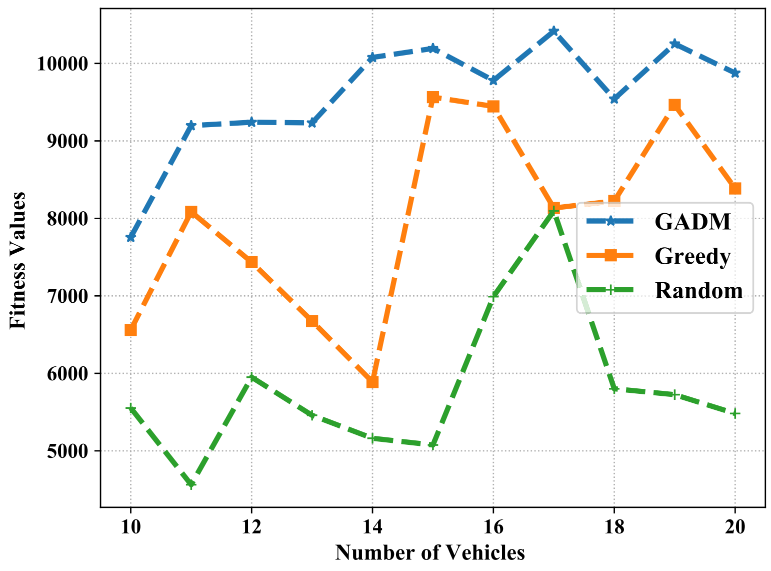

Fig. 10 Fitness values with different number of vehicles (small)

On another hand, Fig. 9 shows the response time of each iteration under different population size. Different from the impacts of *cp* and *cm* on GADM with regards to the response time, the population size has a great impact over the response time of GADM. Generally speaking, the larger the population size, the longer the response time. Considering the high mobility of vehicles in VFC, it is inappropriate to spend much time on decision making. Accordingly, we set the population size to 100 as the default value.

Approach comparison

In addition to the GA involved parameters, other parameters such as n, p, r_i and c_i can also affect the performance of GADM. The number of communities is assumed to be a constant, so the number of vehicles tend to increase in one community when the number of vehicles increase. In this subsection, we have conducted the last set of experiments to evaluate the impact of n over GADM in comparison with other approaches (i.e., "Greedy" and "Random"). The results are shown in Figs. 10 and 11, respectively. The

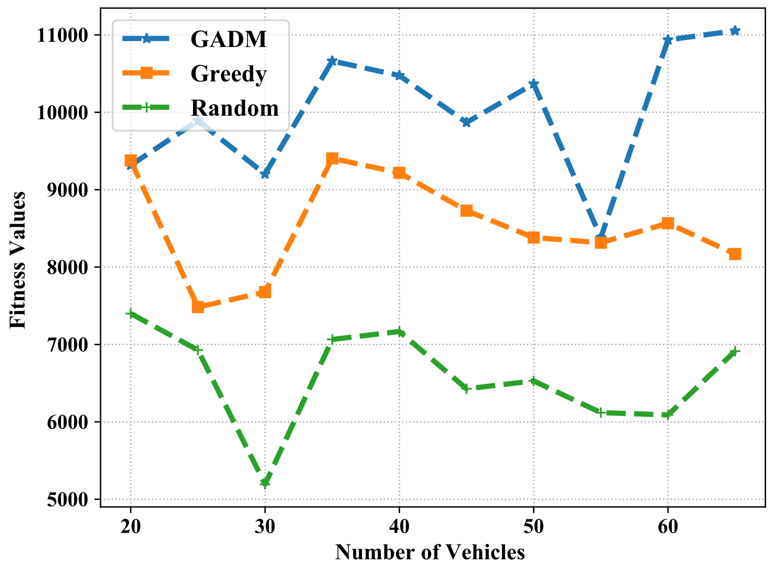

Fig. 11 Fitness values with different number of vehicles (large)

difference between Figs. 10 and 11 lies in that one has small number of vehicles and the other has large number of vehicles. A few observations can be made from both figures as follows.

1) GADM can achieve the best performance, while Random has the worst performance. This observation can be understandable and acceptable, since the random approach makes decision without consideration of any evaluation metric. Random just checks whether the constraint conditions are violated or not.

2) Greedy can achieves the second best performance among the three approaches. Greedy is much better than Random all the time. Different from the random approach, Greedy has a greedy rule (i.e., largest dwell time first) to assist in decision making. Although this greedy rule differs from the fitness function (Eq. (5)), it still brings forth relatively good performance.

3) As the number of vehicles increases, the fitness values do not always increase. For instance, when the number of vehicles equals 16 and 55 respectively, the fitness values decrease compared to the cases with the number of vehicles equal to 15 and 50 respectively. For GADM, the number of vehicles varies, so the length of chromosome varies. Then the entire population is totally different. As discussed earlier, relatively good individuals may be included when the number of vehicles is small. Due to inscrutability of the population, the fitness value can be better when the number of vehicles is small than that when the number of vehicles is large. Another important reason to account for this is that the best fitness value may not be found yet owing to the limited number of iterations. This is because the chromosome length increases when the number of vehicles increases.

4) As for the performance comparison, when the number of vehicles is small, GADM has averagely improved the performance by 20.15% and 65.3% compared the Greedy and Random, respectively. When the number of vehicles is large, GADM has averagely improved the performance by 17.39% and 52.14% compared the Greedy and Random, respectively.

To sum up, GADM outstands other two approaches when the GA involved parameters are set appropriately. Furthermore, considering the mobility of vehicles in VFC, the response time can also be acceptable.

Conclusion

VFC has attracted extensive attention recently due to the proximity of computing resources to the data source. Lots of works have focused on task offloading and resource allocation in VFC from multiple viewpoints. Vehicles as fog nodes can contribute their idle computing resources to the requestors. Meanwhile, they can earn their benefits as computing services are provisioned in a pay-as-you-go model in VFC. We in this paper have proposed to pool the computing resources together to provision services at the edge, with an aim to maximize their own benefits. A GA based algorithm is put forward to make decisions on which communities vehicles can join. The experimental results have shown that GADM indeed is superior to other two approaches.

However, GA based approaches are good at solving the discrete optimization problems at the cost of relatively long response time. Consequently, it is unsuitable to solve the optimization problems in the contexts of VFC, especially when the number of requests is extremely large. Therefore, for the future work, we still plan to design more efficient and effective algorithms to solve the optimization problem.

Acknowledgements
The authors are really grateful to the editor and anonymous reviewers for their professional comments and helpful suggestions.

Authors' contributions
CT came up with the idea and wrote the manuscript. QL partly contributed to the discussion of the work and the correction of the manuscript. SX supervised the creation of manuscript. WC and WF proof-read the manuscript. All authors have approved the manuscript.

Authors' information
Chaogang Tang received his B.S. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2007, 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 currently an assistant professor with the China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, Internet of Things, big data, and WSN.
Shixiong Xia is currently a Professor with the China University of Mining and Technology, Xuzhou, China. His research interests include pattern recognition, intelligent information processing, and information security.
Qing Li is currently a Chair Professor (Data Science) and the Head of the Department of Computing, the Hong Kong Polytechnic University. Formerly, he was the founding Director of the Multimedia software Engineering Research Centre (MERC), and a Professor at City University of Hong Kong where he worked in the Department of Computer Science from 1998 to 2018. Prior to these, he has also at the Hong Kong University of Science and Technology and the Australian National University (Canberra, Australia). Prof. Li served as a consultant to Microsoft Research Asia (Beijing, China), Motorola Global Computing and Telecommunications Division (Tianjin Regional Operations Center), and the Division of Information Technology, Commonwealth Scientific and Industrial Research Organization (CSIRO) in Australia. He has been an Adjunct Professor of the University of Science and Technology of China (USTC) and the Wuhan University, and a Guest Professor of the Hunan University (Changsha, China) where he got his BEng. degree from the Department of Computer Science in 1982.
Weidong Fang received the B.E. degree in industrial electrical automation from Shandong University, Jinan, China, in 1993, the M.E. degree in communication and electronic systems from the China University of Mining and Technology, Beijing, in 1998, and the Ph.D. degree in electromagnetic fields and microwave techniques from Shanghai University, Shanghai, China, in 2016. He is currently an Associate Professor with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai. His current research interest includes energy efficiency and cybersecurity in wireless sensor networks, including trust management, secure network coding, and secure routing protocol.
Wei Chen received the B.Eng. degree in medical imaging and the M.S. degree in paleontology and stratigraphy from the China University of Mining and Technology, Xuzhou, China, in 2001 and 2005, respectively, and the Ph.D. degree in communications and information systems from the China University of Mining and Technology, Beijing, China, in 2008. In 2008, he joined the

School of Computer Science and Technology, China University of Mining and Technology, where he is currently a Professor. His research interests include machine learning, image processing, and computer networks. He is a member of ACM and EAI.

Funding
This work was supported by the Fundamental Research Funds for the Central Universities under Grant 2019XKQYMS46.

Availability of data and materials
No

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer Science and Technology, China University of Mining and Technology, Daxue Road, 221000 Xuzhou, China. [2]The Hong Kong Polytechnic University, Hong Kong, China. [3]Key Laboratory of Wireless Sensor Network & Communication, Shanghai Institute of Micro-system and Information Technology, Chinese Academy of Sciences, 201800 Shanghai, China.

Received: 5 November 2020 Accepted: 29 January 2021
Published online: 24 February 2021

References

1. Perera C, Qin Y, Estrella JC, Reiff-Marganiec S, Vasilakos AV (2017) Fog computing for sustainable smart cities: A survey. ACM Comput Surv 50(3):32–43
2. Wu H, Zhang Z, Guan C, Wolter K, Xu M (2020) Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. IEEE Internet Things J 7(9):8099–8110
3. Wu H, Wolter K, Jiao P, Deng Y, Zhao Y, Xu M (2020) Eedto: An energy-efficient dynamic task offloading algorithm for blockchain-enabled iot-edge-cloud orchestrated computing. IEEE Internet of Things Journal:1–1. https://doi.org/10.1109/JIOT.2020.3033521
4. Hou X, Li Y, Chen M, Wu D, Jin D, Chen S (2016) Vehicular fog computing: A viewpoint of vehicles as the infrastructures. IEEE Trans Veh Technol 65(6):3860–3873
5. Tang C, Wei X, Zhu C, Wang Y, Jia W (2020) Mobile vehicles as fog nodes for latency optimization in smart cities. IEEE Trans Veh Technol 69(9):9364–9375
6. Sookhak M, Yu FR, He Y, Talebian H, Sohrabi Safa N, Zhao N, Khan MK, Kumar N (2017) Fog vehicular computing: Augmentation of fog computing using vehicular cloud computing. IEEE Trans Veh Technol 12(3):55–64
7. Tang C, Zhu C, Wei X, Wu H, Li Q, Rodrigues JJPC (2020) Intelligent resource allocation for utility optimization in rsu-empowered vehicular network. IEEE Access 8:94453–94462
8. Li X, Dang Y, Aazam M, Peng X, Chen T, Chen C (2020) Energy-efficient computation offloading in vehicular edge cloud computing. IEEE Access 8:37632–37644
9. Wu H, Knottenbelt WJ, Wolter K (2019) An efficient application partitioning algorithm in mobile environments. IEEE Trans Parallel Distrib Syst 30(7):1464–1480
10. Rejiba Z, Masip-Bruin X, Marin-Tordera E (2019) Computation task assignment in vehicular fog computing: A learning approach via neighbor advice. In: 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge. pp 1–5
11. Klaimi J, Senouci S, Messous M (2018) Theoretical game approach for mobile users resource management in a vehicular fog computing environment. In: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), Limassol. pp 452–457
12. Xie J, Jia Y, Chen Z, Nan Z, Liang L (2019) Efficient task completion for parallel offloading in vehicular fog computing. China Commun 16(11):42–55
13. Wu H (2018) Multi-objective decision-making for mobile cloud offloading: A survey. IEEE Access 6:3962–3976
14. Guo S, Liu J, Yang Y, Xiao B, Li Z (2019) Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans Mob Comput 18(2):319–333
15. Wu H, Sun Y, Wolter K (2020) Energy-efficient decision making for mobile cloud offloading. IEEE Trans Cloud Comput 8(2):570–584
16. Tang C, Zhu C, Wei X, Chen W, Rodrigues JJPC (2020) Rsu-empowered resource pooling for task scheduling in vehicular fog computing. In: 2020 International Wireless Communications and Mobile Computing Conference (IWCMC), Limassol. pp 1758–1763
17. Wu Q, Ge H, Liu H, Fan Q, Li Z, Wang Z (2020) A task offloading scheme in vehicular fog and cloud computing system. IEEE Access 8:1173–1184
18. Lee S-S, Lee S (2020) Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. IEEE Internet Things J 7(10):10450–10464
19. Wu Q, Liu H, Wang R, Fan P, Fan Q, Li Z (2020) Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems. IEEE Internet Things J 7(1):773–785
20. Peng X, Ota K, Dong M (2020) Multiattribute-based double auction toward resource allocation in vehicular fog computing. IEEE Internet Things J 7(4):3094–3103
21. Liu C, Liu K, Ren H, Zhou Y, Feng L, Guo S, Lee V (2019) Enabling safety-critical and computation-intensive iov applications via vehicular fog computing. In: 2019 15th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN), Shenzhen. pp 378–383
22. Liao H, Zhou Z, Zhao X, Ai B, Mumtaz S (2019) Task offloading for vehicular fog computing under information uncertainty: A matching-learning approach. In: 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), Tangier. pp 2001–2006
23. Ning Z, Huang J, Wang X (2019) Vehicular fog computing: Enabling real-time traffic management for smart cities. IEEE Wirel Commun 26(1):87–93
24. Zhou Z, Liu P, Feng J, Zhang Y, Mumtaz S, Rodriguez J (2019) Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. IEEE Trans Veh Technol 68(4):3113–3125
25. Thakur A, Malekian R (2019) Fog computing for detecting vehicular congestion, an internet of vehicles based approach: A review. IEEE Intell Transp Syst Mag 11(2):8–16

Publisher's Note