International Journal of
Distributed
Sensor Networks

International Journal of Distributed Sensor Networks
2019, Vol. 15(3)
© The Author(s) 2019
DOI: 10.1177/1550147719837859
journals.sagepub.com/home/dsn
SAGE

Joint communication and computing resource allocation in vehicular edge computing

Jianan Sun[1], Qing Gu[2], Tao Zheng[1], Ping Dong[1] and Yajuan Qin[1]

Abstract
The emergence of computation-intensive vehicle applications poses a significant challenge to the limited computation capacity of on-board equipments. Mobile edge computing has been recognized as a promising paradigm to provide high-performance vehicle services by offloading the applications to edge servers. However, it is still a challenge to efficiently utilize the available resources of vehicle nodes. In this article, we introduce mobile edge computing technology to vehicular ad hoc network to build a vehicular edge computing system, which provides a wide range of reliable services by utilizing the computing resources of vehicles on the road. Then, we study the computation offloading decision problem in this system and propose a novel multi-objective vehicular edge computing task scheduling algorithm which jointly optimizes the allocation of communication and computing resources. Extensive performance evaluation demonstrates that the proposed algorithm can effectively shorten the task execution time and has high reliability.

Keywords
Mobile edge computing, vehicular ad hoc network, computation offloading, resource allocation, bat algorithm

Date received: 29 October 2018; accepted: 13 February 2019

Handling Editor: Ming Li

Introduction

As an important part of the intelligent transportation system (ITS), many studies have been devoted to using vehicular ad hoc network (VANET) to improve traffic safety and provide convenience for passengers. The vehicles in VANET are generally equipped with on-board units (OBUs), which provide small-scale computing and storage capacity. However, some emerging applications, such as augmented reality and image processing, require complex computing and large amount of storage.[1] Nowadays, the limited computing and storage resources of vehicles are hard to support such computation-intensive applications. Some researchers have proposed to introduce mobile cloud computing technology into VANET to form vehicular cloud computing (VCC).[2] VCC allows vehicle users to offload their complex computational tasks to the cloud data center, leveraging the rich computing and storage resources of the cloud servers to process tasks. Although VCC effectively reduces the computational load of vehicles, as the quality-of-service (QoS) requirements increase, its defects are becoming more and more obvious.

The powerful cloud data centers are often deployed far away from vehicle users. To access the cloud platform, vehicle users need to traverse the core network through multiple devices such as base stations, hotspots, and routers. Therefore, there is a large delay in

[1]School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China
[2]University of Science and Technology Beijing, Beijing, China

Corresponding author:
Tao Zheng, School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China.
Email: zhengtao@bjtu.edu.cn

the transmission process of the task. However, the service control and data transmission of VANET have high real-time requirements. If the data analysis and control logic are all concentrated in the remote cloud center, it is difficult to guarantee the delay and jitter performance of the applications. In addition, with the rapid increase in the number of users and service types, the data aggregation of massive devices accessing the cloud platform will inevitably cause excessive network load. This leads to problems such as increased delay and severe congestion, which seriously degrade network performance and user experience. Therefore, the computation offloading method of VCC is not suitable for the delay-sensitive task of VANET.

Recently, as a new type of computing model, mobile edge computing (MEC) has become a research hotspot in academia and industry. By extending cloud computing to the edge of the network, MEC can provide high reliability, high bandwidth, and low-latency computing services for mobile devices. This is because the edge server is close to the mobile device, and the user usually only needs a single hop to access the edge cloud, which greatly reduces the communication delay and avoids congestion. In addition, the MEC server at the edge of the network can obtain the user's surrounding environment information in real time. Using these auxiliary information, the server can optimize the services dynamically and rapidly.

With the popularity of applications requiring low latency and situational awareness, MEC technology has been applied in many scenarios, such as augmented reality,[3] smart grid,[4] and network security.[5,6] A significant number of researches have been conducted on the application of MEC in vehicular networks. In the existing researches, MEC servers are usually deployed near infrastructure such as base stations, so their performance will be limited by the infrastructure. First, the spectrum cost of infrastructure is expensive, and vehicular users need to pay for the transmission of their task data. Second, in practice, the infrastructure cannot completely cover vehicles traveling on the road due to the limited communication range and sparse deployment. Third, services cannot be provided when the infrastructure is damaged or user requests exceed its processing capacity. Therefore, it is necessary to study the utilization of computing resources of vehicles on the road to provide a wide range of reliable services. However, the relevant research is still few and there are some difficulties in the implementation. The fast movement of vehicles causes the network topology and wireless channel vary rapidly, which poses challenges to the reliable delivery of computational tasks and the effective collaboration between MEC servers. In addition, the implementation of MEC offloading is composed of data transmission process and task computation process; how to coordinate the communication and computing resources for different vehicle requirements is an important issue affecting the performance of MEC. The main contributions of this article are summarized as follows:

(1) We introduce MEC technology to VANET to build a vehicular edge computing (VEC) system. Efficient and fast computing services are provided by vehicles to overcome the limitations of fixed infrastructures;
(2) A novel multi-objective VEC task scheduling algorithm is proposed, which makes the optimal decision on the allocation of communication and computing resources by observing the state of the VEC system;
(3) Extensive simulations have been conducted, and the results show that the proposed algorithm can effectively shorten the task execution time and has high reliability.

The remainder of this article is organized as follows. Section "Related work" overviews the recent researches on the application of MEC in vehicular networks and investigates their deficiencies. Section "System overview" describes the VEC system and models the computation offloading process in this environment. Then, in section "Optimization solution," we transform the offloading decision problem into a knapsack problem and design a multi-objective VEC task scheduling algorithm based on bat algorithm. The experiments and simulations are presented in section "Experimental evaluation." Finally, we make the conclusion in section "Conclusion."

Related work

Due to its close proximity to mobile users, MEC can effectively reduce the latency of network operations and service delivery. In recent years, many researchers have devoted themselves to applying MEC technology to vehicular networks with high real-time requirements. Liu et al.[7] proposed a software-defined network-enabled vehicular network assisted by MEC, which offers reliable communication services by integrating heterogeneous vehicular networks. The MEC server provides computing resources and supports delay-constrained response to clients, reducing the data transmission time and offering quickly responding service. Huang et al.[8] proposed a 5G-enabled software-defined vehicular network, which leverages MEC servers as local controllers to spread and localize the control plane, monitoring the entire network state and making optimal decision in real time. Huang et al.[9] proposed a distributed reputation management system where MEC servers are used to maintain the reputation of local vehicles. Then, the concept of reputation-assisted optimization is proposed with which service providers optimize resource allocation by referencing reputation values of served vehicles. Huang et al.[10] proposed a software-defined network inside the MEC architecture to resolve the vehicle-to-vehicle (V2V) offloading issues. The MEC server collects the contextual information of vehicles and stores them in the context database. Then, it calculates V2V paths for vehicles that are communicating with each other using the cellular network. Hui et al.[11] proposed an edge computing–based vehicular content dissemination framework. The content provider provides its requirements and uploads the content to the edge computing device (ECD). Then, the ECD evaluates the trajectory, mobility, and traffic status of vehicles and selects the optimal vehicle to relay the content. However, in these researches, MEC servers are deployed only at static infrastructures. It may be too expensive to provide ubiquitous and always-on service using such network architecture.

A few researchers have studied the computation offloading and resource allocation of vehicular networks. The core issue is to offload computation tasks from resource-poor vehicles to resource-rich edge servers. Zhang et al.[12] proposed a cloud-based MEC offloading framework for vehicular networks, which consists of multiple MEC servers and remote clouds. Then, they presented a predictive combination-mode relegation scheme. By predicting the file transmission time and the task execution time, MEC servers can reduce the service latency and network operation cost. Zhang et al.[13] proposed a hierarchical cloud-based offloading framework, where multiple MEC servers can share a backup server to make up for their insufficient computing resources. Then, the authors used a Stackelberg game approach to obtain the optimal offloading strategy, which maximizes the revenues of MEC servers under the delay constraints of computation tasks. Feng et al.[14] proposed a framework named autonomous vehicular edge (AVE) that allows offloading and scheduling without requiring centralized control. The authors also proposed a job caching scheme to better schedule jobs and an ant colony optimization–based scheduling algorithm to assign the jobs. Wu et al.[15] proposed an offloading algorithm based on support vector machine. The algorithm first segments a huge task into several sub-tasks using a weight allocation method. Then, it determines each sub-task to be offloaded or executed locally. Lin et al.[16] proposed a vehicular social edge computing architecture, which inherits the advantages of both edge computing and social network. Then, the authors modeled the system utility and used Lagrangian theory to obtain the optimal resource allocation scheme that maximizes the utility. Du et al.[17] developed a cognitive vehicular network where the IEEE 802.22 channel is reused for computational task offloading. They formulated a dual-side optimization problem to minimize the cost of vehicle terminals and the MEC server. Then, they proposed a online algorithm to tackle the optimization problem using Lyapunov's optimization theory. However, these researches only allocate one resource when performing task offloading, without considering the joint allocation of communication and computing resources.

System overview

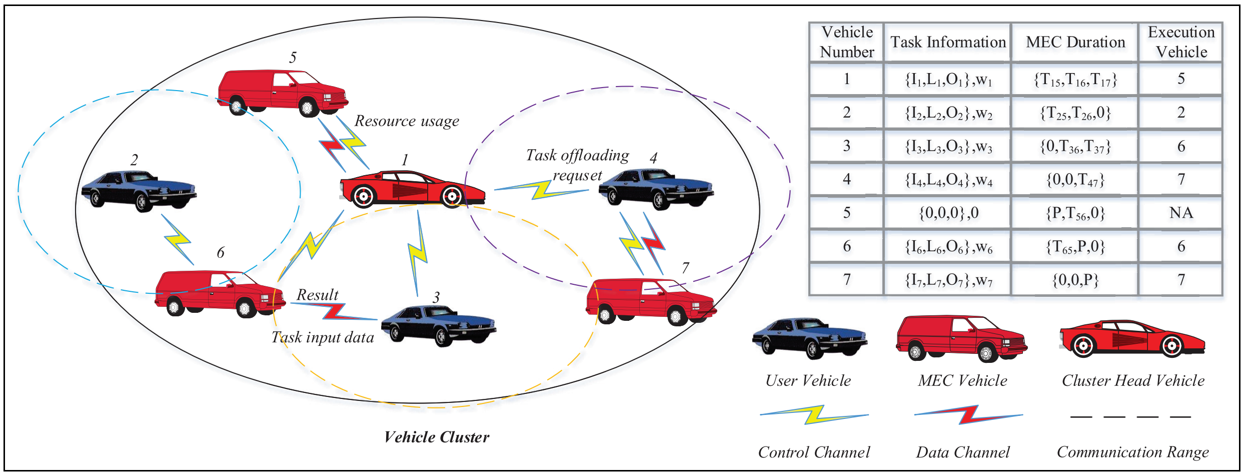
In this section, we first describe the key components of VEC system. Then, we present the communication and computation model, and formulate the offloading decision problem.

System architecture

As shown in Figure 1, this article considers the computation offloading service in VEC network. The high-speed movement of vehicles leads to rapid changes in network topology, resulting in unstable communication and high information exchange overhead. To optimize the data transmission between vehicles and improve the QoS of VEC system, in this research, a group of vehicles moving in the same direction are aggregated, forming a vehicle cluster. The VEC system consists of user vehicles, MEC vehicles, and a cluster head (CH) vehicle.

User vehicle. A user vehicle is equipped with devices such as OBU, Global Positioning System (GPS), and wireless communication module. The OBU can perform some simple tasks, and the GPS enables the user vehicle to obtain its location in real time. With the wireless communication module, the user vehicle can communicate with other vehicles. When the user vehicle moves on the road, it needs to run some new applications, such as augmented reality and image processing. However, the OBU cannot afford the complex computation required by these applications. Thus, the user vehicle generates a requirement for computation offloading.

MEC vehicle. Similar to the user vehicle, a MEC vehicle also has devices such as OBU, GPS, and wireless communication module. In addition, the MEC vehicle is equipped with high-performance MEC server. The MEC server generally has multi-core processors, which can perform multiple tasks at the same time. The MEC vehicle shares its computing resources with other vehicles, allowing user vehicles to offload tasks to the MEC server for better service. When the task computation is complete, the MEC vehicle send the output data back to user vehicles.



Vehicle Number	Task Information	MEC Duration	Execution Vehicle
1	$\{I_1,L_1,O_1\},w_1$	$\{T_{15},T_{16},T_{17}\}$	5
2	$\{I_2,L_2,O_2\},w_2$	$\{T_{25},T_{26},0\}$	2
3	$\{I_3,L_3,O_3\},w_3$	$\{0,T_{36},T_{37}\}$	6
4	$\{I_4,L_4,O_4\},w_4$	$\{0,0,T_{47}\}$	7
5	$\{0,0,0\},0$	$\{P,T_{56},0\}$	NA
6	$\{I_6,L_6,O_6\},w_6$	$\{T_{65},P,0\}$	6
7	$\{I_7,L_7,O_7\},w_7$	$\{0,0,P\}$	7

Figure 1. Vehicular edge computing system topology.

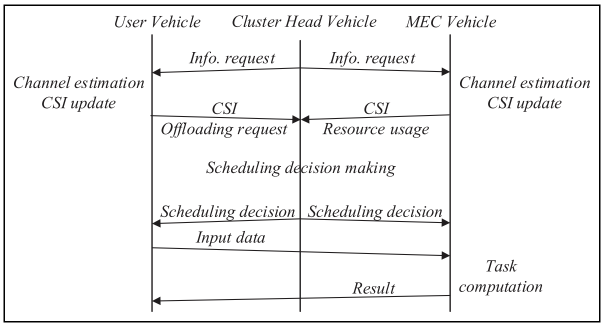


Figure 2. Message flow of VEC system.

CH vehicle. As the manager of VEC system, the CH vehicle controls the computation offloading between user vehicles and MEC vehicles. The CH vehicle collects the task requirements and movement status of user vehicles and allocates the communication resources of the vehicle network and the computing resources of the MEC servers based on these information.

It should be noted that a vehicle in VEC system may have multiple roles at the same time. It may be both a task publisher and a computing resource provider, as well as the scheduling decision-maker. Figure 2 shows the message flow of VEC system, which has the following steps:

(1) The CH vehicle periodically broadcasts its state information through the control channel;
(2) When receiving the signal, other vehicles estimate the quality of the communication channel and update their channel state information (CSI);
(3) Through the control channel, user vehicles send their CSI and task offloading requests to the CH vehicle, and MEC vehicles send their CSI and resource usage to the CH vehicle;
(4) Based on the collected information, the CH vehicle makes the optimal task scheduling decision and broadcasts it to other vehicles through the control channel;
(5) Through the data channel, user vehicles send their task input data to MEC vehicles, and MEC vehicles return the results after performing the tasks.

Communication and computation model

Consider a vehicle cluster scenario as shown in Figure 1, in which there are n vehicles, denoted as a set $N = \{1, 2, \ldots, n\}$. Each vehicle communicates with its neighbor vehicles via wireless channels. Some vehicles run applications with high-intensity computation requirements. According to some existing researches,[18,19] the task of vehicle i can be represented by tuples $T_i = \{I_i, L_i, O_i, T^i_{\max}, w_i\}, i \in N$, where I_i is the size of the input data for computation, L_i is the amount of the computing resources required to accomplish the task, O_i is the size of the output data, and $T^i_{\max}$ is the maximum latency of the task. Considering the different benefits of the tasks to users, a weigh w_i is assigned to each task. In addition, m vehicles in the cluster are equipped with MEC servers. The MEC servers have the same number with the vehicles they belong to, denoted as a set $M = \{S_j | j \in N\}$, and their computation rates are denoted as $C = \{C_j | j \in M\}$.

In VEC system, the CH vehicle regularly monitors the movement status of all vehicles, such as GPS location, speed, and acceleration. Based on these information, the CH vehicle can predict the link duration between vehicles. The link duration between vehicle i and vehicle j is denoted as T_{ij}. Then, the CH vehicle constructs a two-dimensional duration matrix D, in which the value of the element D_{ij} is as follows: if user vehicle i and MEC vehicle j cannot communicate with each other, then $D_{ij} = 0$; else, if $i = j$, which means the MEC vehicle can use its computing resources directly, then $D_{ij} = P$, where P is a very large positive number; if not, $D_{ij} = T_{ij}$.

For each vehicle, its task can be performed locally using the OBU (local computing mode) or offloaded to the MEC server (edge computing mode). Here, we analyze these two task execution modes. Assuming that the OBU computation rate of vehicle i is c_i, the time it takes to perform task T_i locally is $t_{i0} = L_i/c_i$.

If the edge computing mode is adopted, the vehicle needs to send the input data of the task to the MEC server first. When long-term evolution-V (LTE-V)-Direct communication is adopted in VANET, the entire spectrum resource is divided into several resource blocks (RBs) in time and frequency domain.[20] The users in VANET share these RBs. Suppose vehicle i offloads its task to vehicle j, based on the well-known Shannon's formula, the peak transmission rate of vehicle i is

$$\begin{aligned}\eta_{ij}(p_{ij}) = p_{ij} W \log_2 (1 + SINR_{ij}) \\ 1 \leqslant i,j \leqslant N, i \neq j, 1 \leqslant p_{ij} \leqslant kv\end{aligned} \tag{1}$$

where p_{ij} is the amount of RBs allocated to the link between vehicle i and vehicle j, W is the bandwidth of a single RB, and $SINR_{ij}$ is the signal-to-noise ratio of vehicle i received by vehicle j. KV is the total amount of RBs, and each vehicle can use kv RBs at most.

The time consumption of the edge computing mode can be divided into three parts. The first part is the time when the user vehicle transmits the task input data to the MEC vehicle, the second part is the computation time of the task on the MEC server, and the last part is the time that the MEC vehicle returns the task output data to the user vehicle. Since the size of the output data is much smaller than the input data, in this article, we ignore the time to transmit the output data. Based on the above analysis, when vehicle i offloads its task T_i to MEC vehicle j, the time consumption t_{ij} is

$$t_{ij} = \begin{cases} Q, & D_{ij} = 0 \\ \dfrac{I_i}{\eta_{ij}(p_{ij})} + \dfrac{L_i}{C_{ij}}, & 0 < D_{ij} < P \\ \dfrac{L_i}{C_{ij}}, & D_{ij} = P \end{cases} \tag{2}$$

where C_{ij} is the amount of computation rate allocated by vehicle j to perform task T_i, $1 \leqslant C_{ij} \leqslant C_j$. Q is a very large positive number.

Task scheduling analysis

According to the communication and computation model described above, the offloading decisions of different vehicles in VEC system are coupled. When there are multiple vehicles offloading tasks at the same time, they compete with each other for communication resources. Unreasonable resource allocation can result in low data transmission rate and high delay, in which case a vehicle can get the computation result more quickly if it performs the task locally. There is also competition for computing resources between tasks offloaded to the same MEC vehicle. Therefore, a task scheduling scheme that reasonably allocates communication and computing resources has an important impact on the performance of VEC system.

In this article, we consider two optimization objectives: the total execution time of all tasks and the total weight of successful tasks. Here, a successful task is a task whose execution time is less than its maximum latency. Take task T_i as an example, whose maximum latency is $T^i_{\max}$, it can be successfully completed when t_{i0} or $t_{ij} \leqslant T^i_{\max}$. We define a scheduling scheme as the best scheme, if it can minimize the total execution time and maximize the total weight. The scheduling scheme needs to satisfy the following constraints:

(1) *Communication resource constraint.*[19] The resources used for communication in a vehicle cluster are limited. Let p_i denote the amount of RBs allocated to task T_i, and KV denote the total amount of RBs, then $\sum_{i \in N} p_i \leqslant KV$.
(2) *Computing resource constraint.*[16] A MEC server cannot provide computing resources beyond its processing capacity. For MEC server j, its computation rate is C_j, and the computation rate it provides for task T_i is C_{ij}, then $\sum_{i \in N} C_{ij} \leqslant C_j$.
(3) *Delay constraint.* Assuming that vehicle i offloads its task to vehicle j, vehicle i can obtain its computation result only if the task execution time is less than the link duration between the two vehicles, that is, $t_{ij} \leqslant D_{ij}$.

All task scheduling schemes that satisfy the above three constraints are feasible schemes. We define a binary matrix $A = \{a_{ijpq} | a_{ijpq} \in (0, 1)\}_{n \times m \times kv \times C_{\max}}$ as a task scheduling matrix, where $a_{ijpq} = 1$ denotes that vehicle i offloads its task to vehicle j, p RBs are used to transmit the task input data, and computation rate q is used for task computation. $C_{\max}$ is the maximum computation rate that a task can use. Thus, the time consumption of task T_i is

$$t_i = t_{i0} + \sum_{j \in M} \sum_{p=1}^{kv} \sum_{q=1}^{C_{\max}} a_{ijpq}(t_{ij} - t_{i0}) \tag{3}$$

The goal of this article is to find a matrix A under the constraints of communication and computing resources, which can minimize the total execution time and maximize the total weight. Therefore, the task scheduling in VEC system can be formulated as the following combinatorial optimization problem Ω_1

$$\min f_1 = \sum_{i=1}^{n} t_i$$

$$\max f_2 = \sum_{i=1}^{n} w_i u(T_{\max}^i - t_i)$$

$$\text{s.t.}\quad C1 : \sum_{j\in M}\sum_{p=1}^{kv}\sum_{q=1}^{C_{\max}} a_{ijpq} \leqslant 1,\ \forall i \in N$$

$$C2 : \sum_{i=1}^{n}\sum_{j\in M}\sum_{p=1}^{kv}\sum_{q=1}^{C_{\max}} pa_{ijpq} \leqslant KV$$

$$C3 : \sum_{i=1}^{n}\sum_{p=1}^{kv}\sum_{q=1}^{C_{\max}} qa_{ijpq} \leqslant C_j,\ \forall j \in M$$

$$C4 : \sum_{j\in M}\sum_{p=1}^{kv}\sum_{q=1}^{C_{\max}} a_{ijpq}(D_{ij} - t_{ij}) \geqslant 0,\ \forall i \in N$$

where f_1 and f_2 are the objective functions, $u(x)$ is a unit step function, $u(x) = 1$ if and only if $x \geqslant 0$, otherwise $u(x) = 0$. Constraint C1 indicates that there is one and only one execution method for each task. Constraints C2 and C3 represent the communication resource constraint of wireless links and the computing resource constraint of MEC servers, respectively. Constraint C4 represents the delay constraint that the task execution time in edge computing mode should be less than the link duration.

Optimization solution

In this section, we first transform the computation offloading decision problem in VEC system into a knapsack problem. Then, we give a brief introduction of the bat algorithm. Finally, we make many improvements to the bat algorithm and design a multi-objective VEC task scheduling algorithm.

Problem transformation

We regard the wireless link as a big knapsack with a maximum volume of KV and regard the computation rate of MEC server j as a small knapsack with a maximum weight of C_j. Then, the communication and computing resource constraints in VEC system can be seen as a big knapsack containing m small knapsacks. We regard the scheduling decision a_{ijpq} of each task T_i as an item with a value of w_i and a cost of t_{ij}. Furthermore, all scheduling decisions of task T_i are aggregated into a large set. Thus, there are n large sets, and each large set contains $E = m \times kv \times C_{\max} + 1$ items, represented by numbers 1 to E. In this way, the optimization problem Ω_1 is transformed into a knapsack problem Ω_2. The description of Ω_2 is as follows: select an item from each large set and put it into a small knapsack, under the constraints of the small knapsack's weights and the big knapsack's volume, to maximize the total value and minimize the total cost of the items in the big knapsack. The diagram of the knapsack problem is shown in Figure 3.

Obviously, Ω_2 is a multi-dimensional multiple-choice knapsack problem (MMKP) with multiple objectives. MMKP is a kind of NP-hard combinatorial optimization problem. Its exact algorithms (branch and bound method, etc.) have exponential time complexity and are not suitable for online resource allocation. Heuristic algorithms (CGBA,[21] etc.) can obtain suboptimal solution in a short time, suitable for situations with a large number of tasks or resources. However, when the problem has strong constraints, these algorithms are easy to fall into local optimum, and even cannot find a feasible solution. Swarm intelligence algorithm is an effective method for solving combinatorial optimization problems. It has strong global search ability and high convergence rate. Many excellent algorithms have been proposed, and they have been successfully used in solving various practical problems. In order to obtain a good solution in a short time, we design a task scheduling algorithm for VEC system based on the bat algorithm.

Bat algorithm

Bat algorithm was proposed by Yang[22] in 2010. It simulates the behavior of micro-bats in nature, which use echolocation to hunt prey and avoid obstacles. According to the bat's echolocation behavior and its correlation with objective optimization, the parameters and updating equations of n bats during flight are given below.

Let the domain be a D-dimensional search space. Bat i has a random speed v_i, position (solution) x_i, and a pulse frequency f_i. Then, the speed and position are updated according to equations (5) and (6)

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta \tag{4}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x^*)f_i \tag{5}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{6}$$

where $f_{\min}$ and $f_{\max}$ are the minimum and maximum values of the pulse frequency, respectively, and $\beta \in [0, 1]$ is a uniformly distributed random number. x^* is the global best position of the current population.

When hunting prey, a bat emits pulses with high loudness and low emission rate for a wide range of searches. When the prey is found, it reduces the loudness and increases the emission rate to locate the prey more accurately. Equations (7) and (8) describe the update of loudness A_i and emission rate r_i

$$A_i^{t+1} = \alpha A_i^t \tag{7}$$

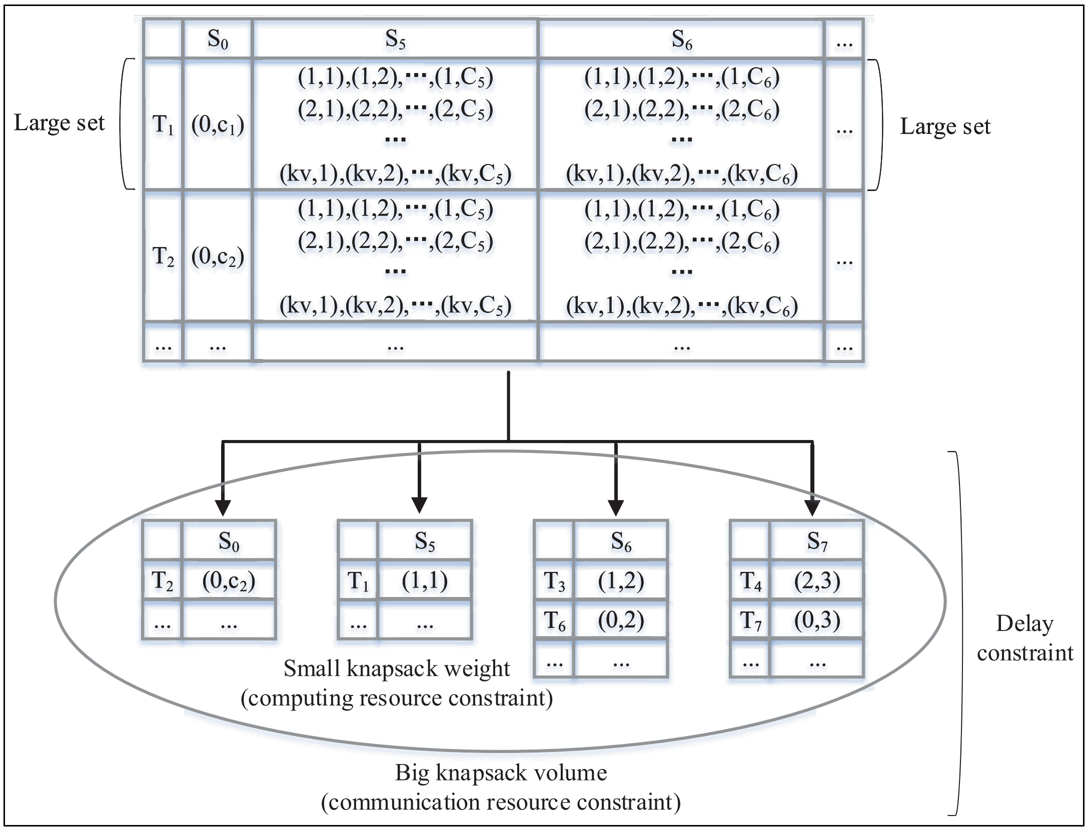


Figure 3. Diagram of the knapsack problem.

$$r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{8}$$

where r_i^0 is the maximum emission rate; α and γ are the attenuation coefficient of loudness and the increasing coefficient of emission rate, respectively, $0<\alpha<1$, $\gamma>0$. Obviously, when $t \to \infty$, $A_i^t \to 0$, $r_i^t \to r_i^0$.

In order to improve the search performance, when a position is determined as the best position of the current population, the new position of each bat is generated in its vicinity

$$x_i^t = x^* + \varepsilon \overline{A^t} \tag{9}$$

where $\varepsilon \in [-1, 1]$ is a D-dimensional random vector and obeys uniform distribution. $\overline{A^t}$ is the average loudness of all bats.

Multi-objective VEC task scheduling algorithm

The bat algorithm can only solve single-objective problems, and it has the disadvantage of being trapped in a local optimum easily. In order to solve the VEC offloading decision problem, we have made the following improvements to the bat algorithm.

Design position update rule. The multi-objective optimization problem does not have a unique global optimal solution, but multiple solutions that are incomparable in terms of global objectives, called Pareto solutions. Thus, there is no population historical best position in the multi-objective bat algorithm. In addition, it is possible that both new and old positions are Pareto solutions in the evolution of bat individuals. For this feature, we adopt the following update rules of bat individuals and population. For a bat individual, when the new position can dominate the old position, it adopts the new position with a certain probability; when neither of them can dominate the other, it randomly selects one of them as the individual's best position. In both cases, the loudness and emission rate of the bat are updated when the new position is selected. For the bat population, the global best position x^* is extended to a Pareto-optimal set S_p. The Pareto solutions in the population are selected and added to S_p. Then, a solution is randomly selected from the set as the bat population's best position. Thus, the update equations (5) and (9) in the bat algorithm should be modified to

$$v_i^{t+1} = v_i^t + (x_i^t - x^p)f_i \tag{10}$$

$$x_i^t = x^p + \varepsilon \overline{A^t} \tag{11}$$

where x^p is a random solution of S_p.

Improve the quality of initial population. The initial distribution of the population has a great influence on the optimization accuracy of the algorithm. In the bat algorithm, the initial bats are generated randomly. Thus, it may happen that the initial bats are clustered together, causing the algorithm to lose its global search ability at the beginning. In this article, the K-means clustering is used to initialize the population, so that the initial bats are evenly distributed in the solution space. For a bat population of size n_p, the initialization steps are as follows:

Step 1. Randomly generate n_p bats as initial cluster centers;
Step 2. Randomly generate n_p bats. For each bat, calculate its distance to each cluster center and assign it into the nearest cluster;
Step 3. Recalculate the center of each cluster;
Step 4. Repeat steps 2 and 3 until the change of the cluster center is less than the given threshold;
Step 5. For each cluster center, select the bat closest to it as the initial bat.

Maintain population diversity. According to the update method of bat algorithm, the bat individuals fly randomly for global search at the beginning of the iteration. When a better position is found, all bats move closer to the better position. As the number of iterations increases, the bat individuals get closer and closer, and their distribution is uneven, resulting in a significant decrease in population diversity. As a result, the bat population over-searches around several better positions, and the algorithm falls into a local optimum. In this article, with reference to non-dominated sorting genetic algorithm II (NSGA-II),[23] an elitist strategy using fast non-dominated sorting and crowding-distance estimation is introduced in the bat algorithm to improve the population diversity.

Fast non-dominated sorting. Fast non-dominated sorting stratifies the population according to the dominance relationship between individuals. For each bat i in the population, it has two parameters n_i and s_i, where n_i is the number of bats dominating bat i and s_i is the set of bats dominated by it. After fast non-dominated sorting, each bat gets a rank r_i. The main steps are as follows:

Step 1. In the population, for each bat i with $n_i = 0$, set its rank r_i to 1, and save it in set H_1;
Step 2. For each bat i in set H_1, obtain its dominating set s_i. Traverse each bat j in s_i, execute $n_j = n_j - 1$, if $n_j = 0$, set bat j's rank r_j to $r_i + 1$, and save it in set H_2;
Step 3. Let $H_1 = H_2$;
Step 4. Repeat steps 2 and 3 until all bats are ranked.

Crowding-distance estimation. The crowding-distance is a measure of how close an individual is to its neighbors. The larger the crowding-distance, the more dispersed the individual distribution, and the better the diversity of the population. Let $f_k(i)$ denote the kth objective function value of bat i. The calculation process for a bat population with l objective functions is as follows:

Step 1. For each objective function, sort the bats in ascending order of function values; set the crowding-distances of bats with maximum value $f_k^{\max}$ and minimum value $f_k^{\min}$ to infinity;
Step 2. Calculate the crowding-distance of bat i using the following equation

$$d_i = \sum_{k=1}^{l} \frac{f_k(i^+) - f_k(i^-)}{f_k^{\max} - f_k^{\min}} \tag{12}$$

where $f_k(i^+)$ and $f_k(i^-)$ are the function values of two bats adjacent to bat i in the kth objective function space.

Elitist strategy. The elite strategy combines the parent population with its offspring population, and the two compete to produce the next generation population. In this way, the predominant individuals in the parent population will not be lost during the evolution process, accelerating the convergence of the algorithm.

After fast non-dominated sorting and crowding-distance estimation, each bat i in the population gets two attributes: rank r_i and crowding-distance d_i. Then, for any two bats i and j, the selection rule is defined as follows: if $r_i \neq r_j$, choose the bat with a lower rank; else, chose the bat with a larger crowding-distance.

According to the above improvements, we design a novel task scheduling algorithm for VEC system based on the bat algorithm. The description of the proposed algorithm is shown in Algorithm 1. The integer coding is used, whereby the position of bat i is denoted as an n-dimensional vector $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$, and each element is an integer between 1 and E. That is, the position in the vector represents the serial number of the large set, and the element value represents the serial number of the item in the large set.

Experimental evaluation

In this section, we conduct a series of experiments and simulations to evaluate the performance of the proposed multi-objective VEC task scheduling algorithm.

Simulation scenario

We simulate the task offloading in VANET using Simulation of Urban Mobility (SUMO) and NS3.

Algorithm 1. Multi-objective vehicular edge computing task scheduling algorithm

```
Input: Task $\{T_i, i \in N\}$, MEC server $\{C_j, j \in M\}$, link duration $T_{ij}$, resource block $KV$
Output: Task scheduling scheme $S_p$
1. Calculate time consumption $t_{i0}$ in local computing mode, and construct duration matrix $D$;
2. Determine the objective functions, and set population size $n_p$ and maximum number of iterations $\lambda$;
3. Initialize $n_p$ bat positions $x_k^0$ using K-means clustering;
4. for $k = 1 \to n_p$ do
5.     Initialize speed $v_k^0$, loudness $A_k^0$, emission rate $r_k^0$;
6.     Generate pulse frequency $f_k$ according to equation (4);
7.     Add bat $k$ to initial population $P_R^0$;
8. Calculate the objective function values of all bats, and select Pareto-optimal set $S_p$;
9. for $t = 0 \to \lambda - 1$ do
10.    Randomly select a solution $x^p$ from $S_p$;
11.    for $k = 1 \to n_p$ do
12.        Update speed $v_k^{t+1}$ according to equation (10);
13.        Generate a random number $rand1$;
14.        if $rand1 \leq r_k^t$ then
15.            Update position $x_k^{t+1}$ according to equation (6);
16.        else
17.            Generate position $x_k^{t+1}$ according to equation (11);
18.        Calculate the objective function values in new positions $x_k^{t+1}$;
19.        Generate a random number $rand2$;
20.        if $rand2 < A_k^t$ and $x_k^{t+1}$ dominate $x_k^t$ then
21.            Accept $x_k^{t+1}$ temporarily to intermediate population $P_I^t$;
22.            Adjust $A_k^{t+1}$ and $r_k^{t+1}$ according to equations (7) and (8);
23.    Combine $P_R^t$ and $P_I^t$ into population $P^{t+1}$;
24.    Reorder $P^{t+1}$ using fast non-dominated sorting and crowding-distance estimation;
25.    Select the first $n_p$ bats to form population $P_R^{t+1}$;
26.    Select bats with rank 1 in $P_R^{t+1}$ to form $S_p$.
```

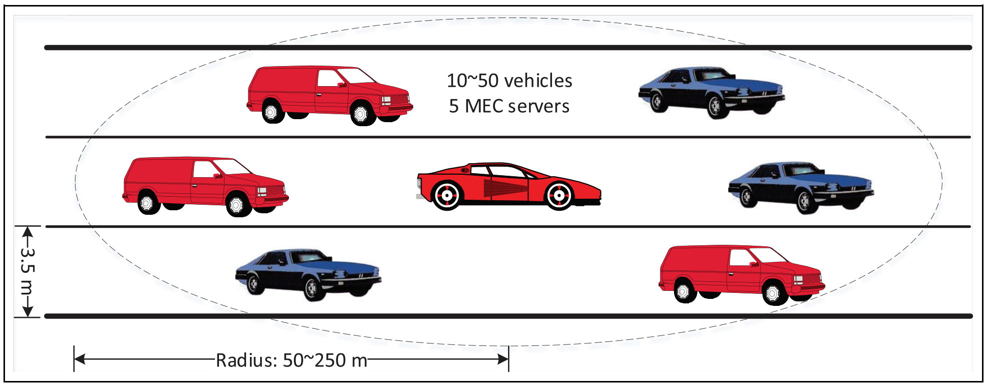


Figure 4. Simulation scenario.

SUMO is used to simulate the movement of vehicles, and NS3 is used to simulate the communication and task offloading between vehicles.

We consider a three-lane highway with a width of 3.5 m per lane, as shown in Figure 4. There is a vehicle cluster on the road, consisting of $n = 10, 15, \ldots, 50$ vehicles with a radius of 50–250 m. These parameters are used to simulate different task loads and vehicle densities in the network. Vehicles are uniformly deployed to lanes, moving at speeds between 60 and 80 km/h. The number of available MEC servers is set to $m = 5$, and their computation rates are 8 GHz.

In the communication simulation, each vehicle has a transmission power of 23 dBm, an antenna gain of 3 dB, and a maximum transmission distance of 320 m. The LTE-V-Direct communication is used, in which a RB occupies 180 kHz of frequency. A bandwidth of 10 MHz at 5.9 GHz is used for our system, so a total of $KV = 50$ RBs are available. Each task can use $kv = 5$ RBs and $C_{\max} = 5$ GHz computation rates at most. The path-loss coefficient is chosen according to a recent research,[24] in which the path loss at 1 m is 47.86 dB and the loss exponent is 2.75. The noise power in the 10-MHz bandwidth is -95 dBm.

Table 1. Simulation parameters for task offloading.

Parameter		Value	Note
n		[10, 50]	Number of vehicles
c_i		[0.5, 1] GHz	OBU computation rate
m		5	Number of MEC servers
C_j		8 GHz	MEC server computation rate
$^*T_i$	l_i	[2, 4] Mb	Input data size
	L_i	1000 cycles/bit	Computing resource requirement
	T^i_{max}	[2, 4] s	Maximum latency
	w_i	[1, 5]	Task weight

OBU: on-board unit; MEC: mobile edge computing.

The simulation parameters for task offloading are presented in Table 1. For comparing the performances, we also consider other three algorithms as follows:

Local computing (LC): all user vehicles compute the tasks using their own OBUs. This scheduling algorithm provides a benchmark for system performance evaluation.

Fairness-driven allocation (FDA): the user vehicle sends an offloading request when the local computation time is longer than the task's maximum latency. Then, the system allocates communication resources equally to all requests. Each request is randomly assigned to a MEC vehicle, and each MEC server allocates its computing resources equally to the assigned requests. Finally, the user vehicle chooses the execution mode with a smaller execution time.

Priority-driven allocation (PDA): similar to the former, the user vehicle sends an offloading request when its OBU cannot provide sufficient computing resources. However, the system allocates communication resources proportionally according to the weight of the tasks, and so does the allocation of the MEC servers'computing resources.

Performance evaluation

Solution precision. We first evaluate the solution precision of the proposed VEC scheduling algorithm. We set the number of vehicles to 30 and the radius of the vehicle cluster to 100 m. Then, the proposed algorithm is run with different population sizes while the maximum number of iterations is 100. Here, we choose the average value of the solutions in the Pareto-optimal set as the result of the proposed algorithm. In addition, we use CPLEX 12.8 to solve the problem model Ω_1 in section "System overview." It should be noted that, in general, the optimal values of objective functions f_1 and f_2 cannot be achieved at the same time. Therefore, we use one objective function each time to solve the optimal

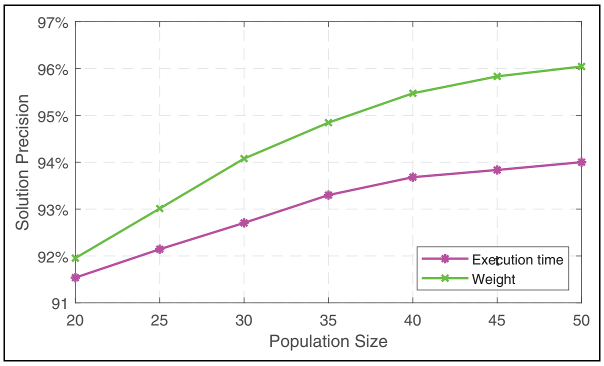


Figure 5. Solution precision of VEC scheduling algorithm.

values of f_1 and f_2, respectively. Let f^* denote the optimal value, then the solution precision of the result f is

$$Precision\,(f) = 1 - \frac{|f - f^*|}{f^*} \tag{13}$$

Figure 5 shows the solution precision of the proposed algorithm with different population sizes. It can be seen that when the population size is small, the proposed algorithm can only reach about 92% of the optimal value. As the population size increases, the solution precision increases, that is, the proposed algorithm can get better solutions. When the population size is 50, the solution precision in terms of execution time and weight is about 94% and 96%, respectively. However, the increase in population size will lead to the increase in solution time. Therefore, we set the population size to 40 in the later experiments.

Performance comparison. In this section, we evaluate the performance of four task scheduling algorithms in given scenarios. Since the load and weight of the tasks vary in different scenarios, we use the average execution time of all tasks and the weight ratio of successful tasks as the evaluation metrics. In addition, we compare the task offloading ratio of different algorithms.

In scenario 1, we fix the radius of the vehicle cluster to 100 m and vary the number of vehicles from 10 to 50. Then, the average execution time, weight ratio, and offloading ratio are shown in Figures 6–8, respectively. It can be observed that the offloading algorithm has obvious advantages over the local computing algorithm. When using LC algorithm, each vehicle has only a small amount of computing resources to perform its task. Thus, it is difficult for vehicles to accomplish tasks with large computational loads within a specified time. As a result, LC has the longest average execution time (about 4.20 s) and the smallest weight ratio (about 8.97%). While using the offloading algorithm, user vehicles can offload the complex computational tasks to MEC vehicles, leveraging the rich computing resources to process tasks. Therefore, the number of

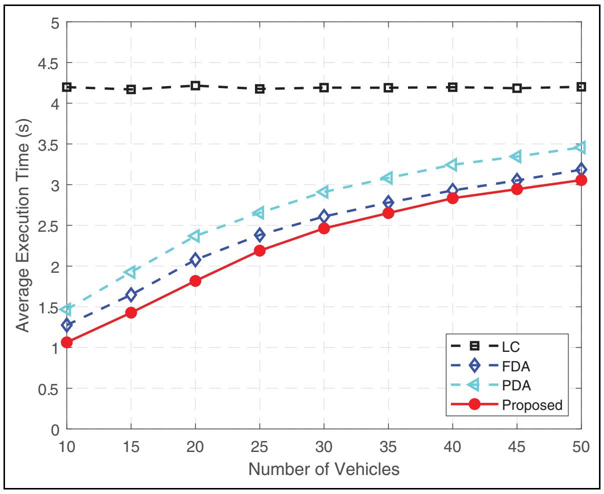


Figure 6. Average execution time for scenarios with different number of vehicles.

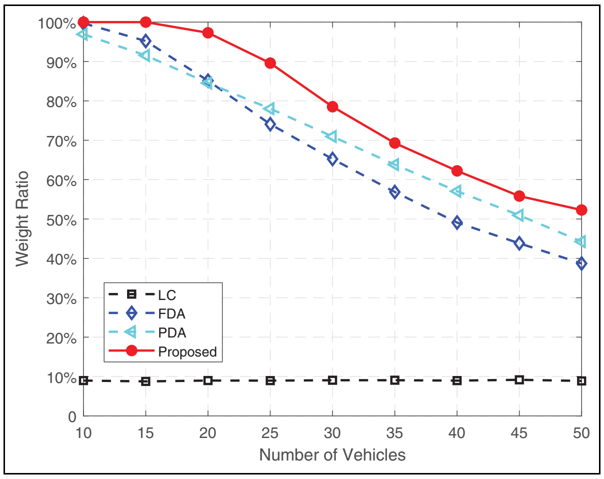


Figure 7. Weight ratio for scenarios with different number of vehicles.

successful tasks increases, resulting in shorter average execution time and larger weight ratio.

From Figures 6 and 7, it can be observed that as the number of vehicles increases, the average execution time of the offloading algorithm increases while the weight ratio decreases. The more users compete for resources, the less resources each user can obtain, and the longer it takes to accomplish the tasks. PDA has a longer average execution time than FDA, but it can achieve a larger weight ratio. This is because PDA tends to allocate resources to users with large weights, making it easier for them to accomplish tasks within delay constraints. The proposed algorithm considers both task execution time and weight and optimizes the allocation of communication and computing resources. Compared with other algorithms, it can obtain the shortest average execution time and the largest weight ratio at the same time.

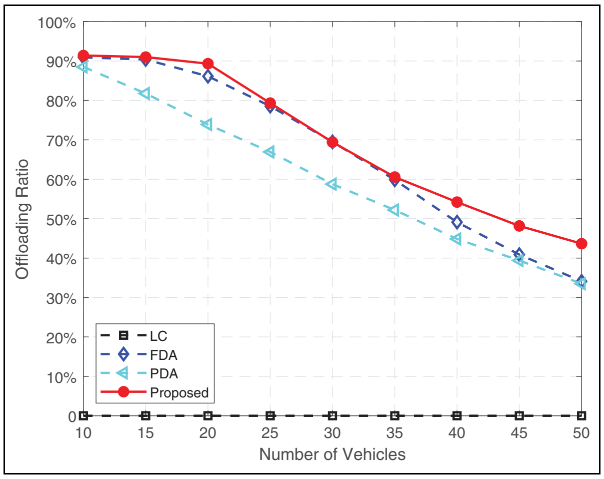


Figure 8. Offloading ratio for scenarios with different number of vehicles.

In Figure 8, as the number of vehicles increases, the offloading ratio of the offloading algorithm decreases gradually. This is because more and more users cannot obtain shorter task execution time through the edge computing mode and instead choose to perform their tasks locally. When the number of vehicles is small, FDA and the proposed algorithm have similar offloading ratios, which are greater than that of PDA. However, the offloading ratio of the proposed algorithm decreases slowly when the number of vehicles is larger than 35. This shows that the proposed algorithm can make the most of the edge computing resources by optimizing resource allocation.

In scenario 2, the number of vehicles is kept at 30, while the cluster radius varies from 50 to 250 m. Then, the average execution time, weight ratio, and offloading ratio are shown in Figures 9–11, respectively. Similarly, LC algorithm has the worst performance. With the increase in cluster radius, the average execution time of the offloading algorithm increases and the weight ratio decreases, while the metrics of LC remain unchanged. This is because the task execution time of the offloading algorithm includes the data transmission time and the computation time on MEC servers. The increase in cluster radius reduces the signal-to-noise ratio between vehicles, resulting in decreased transmission bandwidth and increased transmission time. LC does not involve the data transmission process, and its task execution time is only the local computation time.

In Figures 8 and 9, FDA has a better average execution time, while PDA has a better weight ratio. The proposed algorithm takes into account the link duration, wireless bandwidth, and computing resource status, providing high-rate wireless channels for users to transmit tasks. Thus, it has the best performance.

In Figure 11, with the increase in cluster radius, the offloading ratio of the offloading algorithm decreases.

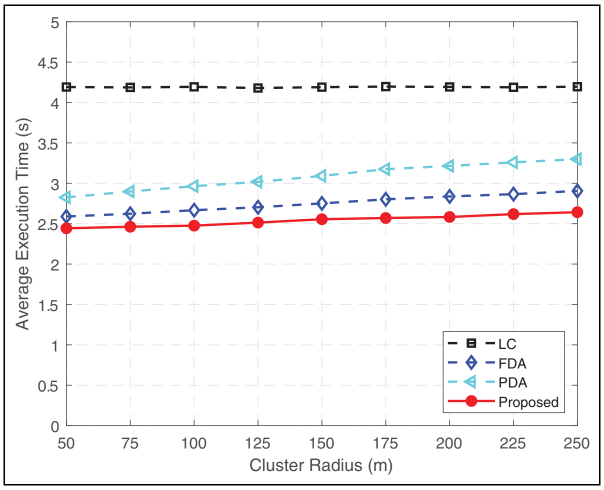


Figure 9. Average execution time for scenarios with different cluster radius.

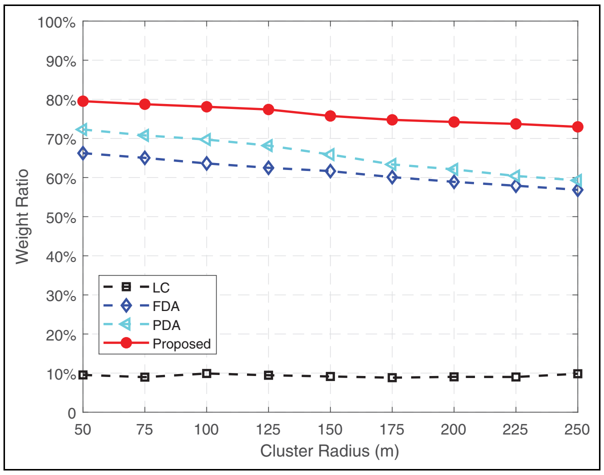


Figure 10. Weight ratio for scenarios with different cluster radius.

A similar conclusion is obtained that the FDA and the proposed algorithm perform better than PDA. When the cluster radius is larger than 125 m, the benefit of the proposed algorithm is highlighted.

In scenario 1, the increase in the number of vehicles leads to significant changes in the evaluation metrics of the offloading algorithm. While in scenario 2, these evaluation metrics change slowly as the cluster radius increases. This shows that the task load has a greater impact on the performance of the offloading algorithm than the communication environment.

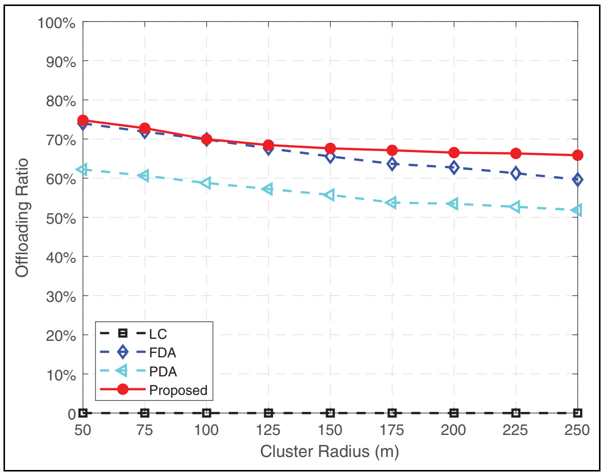


Figure 11. Offloading ratio for scenarios with different cluster radius.

Conclusion

In order to meet the low latency and situational awareness requirements of emerging applications, MEC technology has been applied in vehicular networks. This article introduces MEC technology to VANET to build a VEC system, which provides a wide range of reliable services by utilizing the computing resources of vehicles on the road. Then, the computation offloading decision problem in this system is studied, and a novel multi-objective VEC task scheduling algorithm is proposed, by jointly optimizing the allocation of communication and computing resources. Extensive simulations have been conducted, and the results demonstrate that the proposed algorithm can effectively shorten the task execution time and has high reliability.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported in part by the Fundamental Research Funds for the Central Universities (grant no. 2017YJS031), the Beijing Municipal Natural Science Foundation of China (grant no. 4182048), the National Natural Science Foundation of China (grant no. 61872029), and the National Basic Research Program of China (973 Program; grant no. 2013CB329101).

ORCID iDs

Jianan Sun https://orcid.org/0000-0003-0490-056X
Tao Zheng https://orcid.org/0000-0002-1677-6466

References

1. Lin F, Zhou Y, Pau G, et al. Optimization-oriented resource allocation management for vehicular fog computing. *IEEE Access* 2018; 6: 69294–69303.

2. Whaiduzzaman M, Sookhak M, Gani A, et al. A survey on vehicular cloud computing. *J Netw Comput Appl* 2014; 40: 325–344.
3. Al-Shuwaili A and Simeone O. Energy-efficient resource allocation for mobile edge computing-based augmented reality applications. *IEEE Wirel Commun Le* 2017; 6(3): 398–401.
4. Kumar N, Zeadally S and Rodrigues J. Vehicular delay-tolerant networks for smart grid data management using mobile edge computing. *IEEE Commun Mag* 2016; 54(10): 60–66.
5. Lin F, Zhou Y, An X, et al. Fair resource allocation in an intrusion-detection system for edge computing: ensuring the security of internet of things devices. *IEEE Consum Electron Mag* 2018; 7(6): 45–50.
6. An X, Zhou X, Lü X, et al. Sample selected extreme learning machine based intrusion detection in fog computing and MEC. *Wirel Commun Mob Com* 2018; 2018: 7472095.
7. Liu J, Wan J, Zeng B, et al. A scalable and quick-response software defined vehicular network assisted by mobile edge computing. *IEEE Commun Mag* 2017; 55(7): 94–100.
8. Huang X, Yu R, Kang J, et al. Exploring mobile edge computing for 5G-enabled software defined vehicular networks. *IEEE Wirel Commun* 2017; 24(6): 55–63.
9. Huang X, Yu R, Kang J, et al. Distributed reputation management for secure and efficient vehicular edge computing and networks. *IEEE Access* 2017; 5: 25408–25420.
10. Huang C, Chiang M, Dao D, et al. V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture. *IEEE Access* 2018; 6: 17741–17755.
11. Hui Y, Su Z, Luan TH, et al. Content in motion: an edge computing based relay scheme for content dissemination in urban vehicular networks. *IEEE T Intell Transp*. Epub ahead of print 13 November 2018. DOI: 10.1109/TITS.2018.2873096.
12. Zhang K, Mao Y, Leng S, et al. Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading. *IEEE Veh Technol Mag* 2017; 12(2): 36–44.
13. Zhang K, Mao Y, Leng S, et al. Optimal delay constrained offloading for vehicular edge computing networks. In: *Proceedings of IEEE international conference on communications (ICC)*, Paris, 21–25 May 2017, pp.1–6. New York: IEEE.
14. Feng J, Liu Z, Wu C, et al. AVE: autonomous vehicular edge computing framework with ACO-based scheduling. *IEEE T Veh Technol* 2017; 66(12): 10660–10675.
15. Wu S, Xia W, Cui W, et al. An efficient offloading algorithm based on support vector machine for mobile edge computing in vehicular networks. In: *Proceedings of 2018 10th international conference on wireless communications and signal processing (WCSP)*, Hangzhou, China, 18–20 October 2018, pp.1–6. New York: IEEE.
16. Lin F, Lü X, You I, et al. A novel utility based resource management scheme in vehicular social edge computing. *IEEE Access* 2018; 6: 66673–66684.
17. Du J, Yu R, Chu X, et al. Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. *IEEE T Veh Technol* 2019; 68: 1079–1092.
18. Mao Y, Zhang J and Letaief K. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE J Sel Area Comm* 2016; 34(12): 3590–3605.
19. Wang C, Yu F, Liang C, et al. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing. *IEEE T Veh Technol* 2017; 66(8): 7432–7445.
20. 3GPP TS 36.211 v14.2.0:2017. Evolved universal terrestrial radio access (E-UTRA): physical channels and modulation.
21. Cherfi N and Hifi M. A column generation method for the multiple-choice multi-dimensional knapsack problem. *Comput Optim Appl* 2010; 46(1): 51–73.
22. Yang X. A new metaheuristic bat-inspired algorithm. In: González JR, Pelta DA, Cruz C, et al. (eds) *Nature inspired cooperative strategies for optimization (NICSO)*, Vol. 284. Berlin: Springer, 2010, pp.65–74.
23. Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T Evolut Comput* 2002; 6(2): 182–197.
24. Bazzi A, Masini BM, Zanella A, et al. On the performance of IEEE 802.11p and LTE-V2V for the cooperative awareness of connected vehicles. *IEEE T Veh Technol* 2017; 66(11): 10419–10432.