Journal on Internet of Things
DOI:10.32604/jiot.2020.011792
Article

An Optimization Scheme for Task Offloading and Resource Allocation in Vehicle Edge Networks

Yuxin Xu[1], Zilong Jin[1,2,*], Xiaorui Zhang[1] and Lejun Zhang[3]

[1]School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[2]Jiangsu Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[3]College of Information Engineering, Yangzhou University, Yangzhou, 225127, China
[*]Corresponding Author: Zilong Jin. Email: zljin@nuist.edu.cn
Received: 20 May 2020; Accepted: 15 August 2020

Abstract: The vehicle edge network (VEN) has become a new research hotspot in the Internet of Things (IOT). However, many new delays are generated during the vehicle offloading the task to the edge server, which will greatly reduce the quality of service (QOS) provided by the vehicle edge network. To solve this problem, this paper proposes an evolutionary algorithm-based (EA) task offloading and resource allocation scheme. First, the delay of offloading task to the edge server is generally defined, then the mathematical model of problem is given. Finally, the objective function is optimized by evolutionary algorithm, and the optimal solution is obtained by iteration and averaging. To verify the performance of this method, contrast experiments are conducted. The experimental results show that our purposed method reduces delay and improves QOS, which is superior to other schemes.

Keywords: Vehicle edge network; evolutionary algorithm; task offloading; resource allocation

1 Introduction

Recently, the IOT is a research hotspot of academic and industrial sessions. It consists of all things in life that can connect to the internet, such as mobile phones, laptops, vehicles, etc., with limited CPU capacity and energy characteristics [1]. Vehicle network, as one of the important branches, has become a new research field. With the advent of the 5G era, virtual reality technology (VR), automatic driving and other applications in vehicle network have put forward new requirements for vehicle communication with low delay and low energy consumption [2]. Previous studies have focused on transferring data to cloud computing centers for data processing and feedback of results [3]. Although it takes less time than local processing, the process of data transfer creates additional delays because the physical distance in the cloud center is too remote for mobile devices. MEC technology can solve this problem very well, it transfers the computation of data from the center of cloud computing to the edge of mobile network, and is closer to the vehicle in physical distance [4,5]. By MEC technology, the vehicle can bypass the Internet delay, which can reduce the computation delay and transmission delay. However, the computing power of the edge server is also limited in reality, and it cannot support the computing tasks of all mobile users in the region [6,7]. Therefore, the reasonable allocation of computing resources in the edge server is very important for delay-sensitive applications. In the vehicle edge network, this dynamic task offloading scheme is an effective way to reduce the delay of the network. It is decided by the mobile user according to the requirement of service quality and the computing resources of the edge server, that is, either offload the computing task to the edge server for task execution, or the task will be done locally in the mobile device.

Although task offloading is an effective solution for delay-sensitive applications such as automatic

driving. However, issues such as which tasks are offloaded to the edge server and which edge servers are selected to perform the offloaded tasks have not been resolved. How to allocate computing resources of edge servers effectively becomes a new challenge. In 2014, a threshold strategy-based service migration mechanism in mobile micro-clouds was proposed [8]. The problems of service performance and service migration became important when users move to the coverage areas of different base stations. They modeled the problem as a Markov decision process (MDP) and demonstrated that the optimal strategy for service migration was a threshold strategy when mobile users follow a random walk mobile model. Aiming at the problem of computing offloading decision among mobile device users, the problem of decentralized computing offloading decision among mobile users as a decentralized computing offloading game was described [9]. This method can realize the Nash equilibrium of the game and concretize the efficiency ratio of the concentrated optimal solution, and finally realize the efficient computing unloading of mobile cloud computing.

Based on the current research situation, an evolutionary algorithm-based task offloading and resource allocation scheme, which is used for increasing the delay caused by the unreasonable task offloading and resource allocation, is proposed in this paper. First, we concretize all offloading delay in the vehicle edge network, then the total delay is mathematically modeled. Finally, the evolutionary algorithm is used to optimize the target problem by initializing population, calculating fitness, selecting, crossover and mutation. And then the optimal solution can be obtained after 100 iterations. Based on the real-time task and resource situation, the optimal proportion of task offloading and the optimal scheme of resource allocation can be obtained finally.

This paper is organized as follows: In Section 2, related work will be discussed. Section 3 describes the network model and problem definition. And our proposed method is introduced elaborately in Section 4. Section 5 gives the experimental results of the proposed scheme. Finally, a conclusion is given in Section 6.

2 Related Work

In edge networks, task offloading and computing resource management have been the focus of research in mobile edge computing. Liu et al. [10] studied the computational resource management problem of delay sensitive class applications in mobile edge networks. They jointly considered the allocation of radio and computational resources to achieve the goal of reducing service delays. First, the delay of the service requested by the mobile user is modeled, including wireless delay, network delay and computing delay:

$$T_k = \frac{d_k}{R_k} + \sum_{n \in N} a_{k,n}(\tau_{k,n} + \frac{c_k}{f_{k,n}}) \tag{1}$$

where d_k is the task size of the user k, R_k is the data transfer rate. $a_{k,n} \in (0,1)$ represents whether mobile user k transfer a task to MEC Server n. $\tau_{k,n}$ is the network delay between mobile user k and MEC Server n. c_k represents the computing resources needed to complete the task of user k. $f_{k,n}$ is the computing resources allocated by MEC Server n.

Then, a delay minimization problem that jointly considers uplink transmission power, task allocation, and computational resource allocation is formed. Finally, using the proposed joint radio and computational resource management (iRAR) algorithm, the delay problem is optimized and its advantages are verified by a large number of simulations.

Luo et al. [11] focused on reducing energy consumption for task unloading in MEC. The authors first consider the energy consumption generated by the interaction between tasks and perform mathematical modeling. Then, the task execution flow is identified by computing k-hop connectivity, so that a directed graph is constructed based on the task interaction matrix and the delay of each task execution flow is formulated. Finally, the energy consumption minimization problem is expressed as a quadratic constrained

integer mixed quadratic programming problem, and an effective heuristic method is proposed.

Although many scholars have studied resource allocation schemes in mobile edge networks. However, they only consider the time consumption of task offloading or the additional energy consumption caused by the process. Moreover, there are only two options for mobile user's task uninstall, all transfer and no transfer, ignoring the computing power of mobile user itself.

3 System Model and Problem Definition

3.1 Network Model

We consider a vehicle edge computing system, as shown in Fig. 1, in which i vehicles are randomly distributed in the communication range of j MEC servers, which are connected by optical fiber between them. All vehicles in the system need to run computing-intensive and delay-sensitive services with the help of MEC servers. Each vehicle can only establish a connection with one MEC server and offload the task to that MEC server. When a vehicle is in multiple MEC server communication range at the same time, the vehicle needs to select the nearest MEC server to establish a connection according to the distance between the vehicle and the server.

$$d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{2}$$

where (x_i, y_i) is the location information of the vehicle i, (x_j, y_j) the location information of the MEC server j.

Each vehicle has computing tasks, which can be processed locally or unloaded to a MEC server for execution. We denote the unloading ratio of the task j of the vehicle i as $x_{i,j} \in [0,1]$, 0 as not transferred to the MEC server, and 1 as all tasks transferred to the MEC server for execution. We assume that in the vehicle edge computing system, the MEC server has sufficient computing resources to complete the task of vehicle publishing, and does not consider the energy consumption of the MEC server.

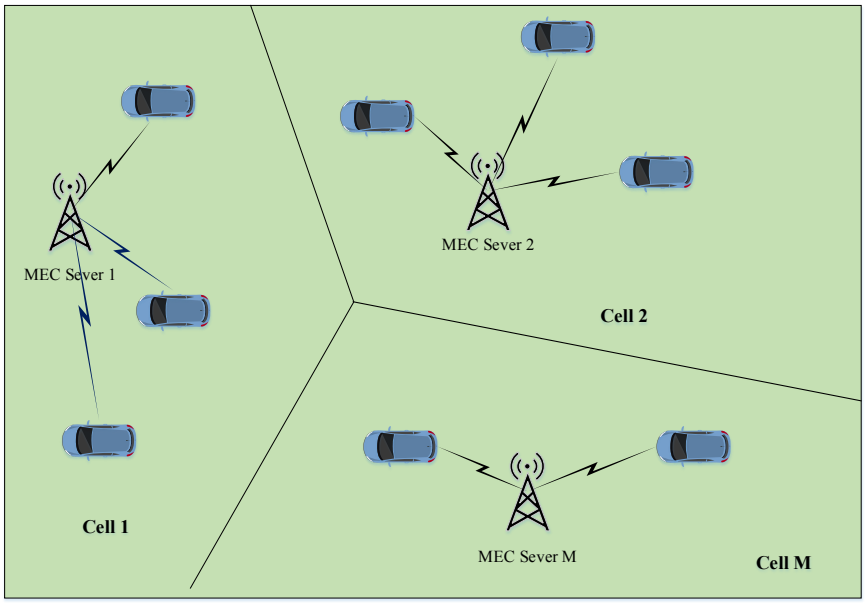

Figure 1: Vehicle edge computing system

3.2 Cost of Local Model

Denote the data size of the task, the highest tolerance delay as $D_{i,j}$ (bit), σ_j (s) respectively. For task

j, the consumption of completing the task is mainly composed of two parts: 1. The energy consumption required to complete the task 2. The time taken to complete the task. For the local computing module, when $x_{i,j} \neq 1$, the remaining $1 - x_{i,j}$ tasks will be done locally.

3.2.1 Local Execution Delay

For the partly task performed locally, the latency is generated from calculating the task:

$$T_{i,j}^{L} = \frac{(1 - x_{i,j}) \cdot D_{i,j} \cdot C_k}{f_i^L} \tag{3}$$

where C_k (cycles/bit) represents the number of CPU cycles of 1 bit data, and f_i^L (cycles/second) represents the computing capacity of the vehicle *i*.

3.2.2 Local Execution Energy Consumption

The local computational energy consumption is generated by the vehicle *i* performing the remaining tasks. We use $E_{i,j}^{L}$ to represent the local computational energy consumption corresponding to the task *j*:

$$E_{i,j}^{L} = (1 - x_{i,j}) \cdot D_{i,j} \cdot C_k \cdot Z_i \tag{4}$$

where Z_i (J/cycle) is the energy consumption per CPU cycle.

3.3 Cost of MEC Model

The whole offloading process consists three steps if the vehicle *i* decides to offload part of the task to the MEC server for execution: 1. The vehicle *i* offloads the relevant data of the task to the MEC server *j* (data size, maximum tolerance delay). 2. MEC server *j* assign computing resources to execute tasks. 3. MEC server *j* feedback the result of the task to the vehicle *i*. Hence, for vehicle *i*, delay and energy consumption are generated by these three processes.

3.3.1 Latency on MEC Sever

The delay of offloading a task to the MEC server is generated by the following three processes: offload the task, MEC server executes the task, and MEC server feedbacks the result of the task.

Latency of Offloading Task

According to the first step, when the vehicle decides to offload part of the task to the MEC server, it needs to upload the relevant data to the MEC server, and the delay of the task offloading is expressed as $T_{i,j}^{o}$:

$$T_{i,j}^{o} = \frac{x_{i,j} \cdot D_{i,j}}{r_{i,j}^{up}} \tag{5}$$

where $r_{i,j}^{up}$ represents the data transfer rate of the uplink between the vehicle *i* and the MEC server *j*, which is expressed as:

$$r_{i,j}^{up} = B_{i,j} \log_2(1 + SINR) \tag{6}$$

where $B_{i,j}$ is the allocated channel bandwidth and *SINR* is the signal-to-noise ratio. *SINR* can be obtained by the following formula:

$$SINR=\frac{P_{i.j}^{T}\cdot O_{i,j}}{P_Z+\sum_{u=1}^{i}P_{u,j}^{T}\cdot O_{u,j}} \tag{7}$$

where $P_{i,j}^{T}$ is the transmission power, $O_{i,j}$ is the channel gain between the vehicle *i* and the MEC server *j*, which can be expressed as:

$$O_{i,j}=d_{i,j}^{-\alpha} \tag{8}$$

where α is the path loss factor and here is set to -4 [12].

Latency of Offloading Computing

For the computing delay in the second step, the MEC server will assign a certain computing resource to execute the task after receiving the task by the vehicle, which can be expressed as:

$$T_{i,j}^{M}=\frac{x_{i,j}\cdot D_{i,j}\cdot C_k}{f_j^{M}} \tag{9}$$

Where f_j^{M} is the computing resource assigned by the MEC server to the task *j*, and its size is limited by the total computing resource of the MEC server f_M :

$$\sum_{j=1}^{J}f_j^{M}\le f_M \tag{10}$$

Latency of Result Feedback

In the last step, the MEC server *j* will feedback the task execution results to the vehicle *i*, and the time required is the time the vehicle *i* receives the task results from the downlink, which is expressed as:

$$T_{i,j}^{d}=\frac{\varepsilon\cdot x_{i,j}\cdot D_{i,j}}{r_{i,j}^{d}} \tag{11}$$

where ε is the ratio of output data to input data, here we set to 0.1 [13–15]. $r_{i,j}^{d}$ is the data transmission rate of the downlink. For vehicles *i*, we assume that the transmission rate of the uplink is the same as the downlink.

Overall, we represent the total time MEC server to complete the task as follows:

$$T_{i,j}^{MEC}=T_{i,j}^{o}+T_{i,j}^{M}+T_{i,j}^{d} \tag{12}$$

3.3.2 Energy Consumption on MEC Sever

In the vehicle edge network, we do not consider the energy consumption of MEC servers. So for vehicles *i*, we divide the process of generating energy consumption into three steps, specifically: 1. Energy consumption for offloading task data 2. Energy consumption for task calculation 3. Energy consumption for receiving feedback

Energy Consumption of Offloading Task

Firstly, after the vehicle *i* decides to offload part of the task to the MEC server, the relevant data needs to be uploaded to the MEC server, and the energy consumption of this part can be expressed as $E_{i,j}^{up}$:

$$E_{i,j}^{up}=x_{i,j}\cdot D_{i,j}\cdot W_{i,j} \tag{13}$$

where $W_{i,j}$ is the transmission energy consumption of a bit data.

Energy Consumption of Local Computing

Secondly, when the vehicle i upload part of the task to the MEC server, the remaining tasks need to be performed locally, and the energy consumption of this part has been stated in 3.2.2, which is no longer stated here.

Energy Consumption of Receiving Feedback

Finally, MEC server j need to feedback the task result to the vehicle i after executing the task, and the vehicle needs to consume energy to receive the task result, which is expressed as:

$$E_{i,j}^{d} = \alpha \cdot x_{i,j} \cdot D_{i,j} \cdot W_{i,j} \tag{14}$$

Above all, the total energy consumption of the vehicle i to complete the task is generated by the above three processes. We express the total energy consumption as:

$$E_{i,j} = E_{i,j}^{up} + E_{i,j}^{L} + E_{i,j}^{d} \tag{15}$$

3.4 Problem Formulation

In our proposed vehicle edge network system, we consider the delay and energy consumption cost synthetically. The total system cost is defined as the weighted sum of total energy consumption and delay, and a new cost function is proposed:

$$C_{i,j} = \lambda \cdot T_{i,j} + \mu \cdot E_{i,j} \tag{16}$$

where $\lambda+\mu=1$, λ and μ are the weights of delay and energy consumption, respectively. The values of λ and μ can be adjusted according to the type of service and the remaining energy of the vehicle. For example, when the remaining energy of the vehicle is less than 20%, the value of μ will be much bigger than λ. In our system, the task we study is delay-sensitive services such as automatic driving, so the requirement for delay is higher, λ is set to 0.8 [16–19].

The delay of task needs to consider the time of execution of local tasks and the time of execution of tasks offloaded to the MEC. The maximum value of both is the total time spent by the task:

$$T_{i,j} = \max\left\{T_{i,j}^{L}, T_{i,j}^{MEC}\right\} \tag{17}$$

Our goal is to minimize the total system cost by jointly optimizing the task offloading decision of the vehicle i and the resource allocation strategy of the MEC server. In addition, the constraints of limited communication and computing resources need to be considered. We define this joint optimization problem as:

$$\begin{aligned}
&\mathrm{P1}: \min_{\left\{x_{i,j}, B_{i,j}, f_j^M\right\}} \sum_{i}^{I}\sum_{j}^{J}\left\{\lambda \cdot \max\left\{T_{i,j}^{L}, T_{i,j}^{MEC}\right\} + \mu \cdot E_{i,j}\right\}1 \\
&s.t.\quad C1): x_{i,j} \in [0,1], \forall i \in I \\
&\qquad C2): \max\left\{T_{i,j}^{L}, T_{i,j}^{MEC}\right\} \le \sigma_j, \forall j \in J \\
&\qquad C3): 0 \le \sum_{j}^{J} f_j^M \le f_M, \forall j \in J \\
&\qquad C4): 0 \le \sum_{i}^{I} B_{i,j} \le B_T, \forall \mathrm{i} \in \mathrm{I}
\end{aligned} \tag{18}$$

where the constraint *C1* indicates that the value of the offload strategy is a value between 0 and 1. Constraint *C2* ensures the delay constraint of the computational task, and σ_j indicates the highest tolerance delay. Constraint *C3* shows that the sum of the computing resources allocated by all tasks cannot exceed the computing resources of the MEC server. And constraint *C4* ensures that the sum of bandwidth allocated by all tasks cannot exceed the total channel bandwidth.

4 Proposed Method

P1 is a single-objective optimization function. Our goal is to find optimal task offloading and MEC server resource allocation strategies to minimize the total cost. However, as the dimension of the variables increases, the computational complexity will increase dramatically. To solve this optimization problem, we propose an optimal task offloading and resource allocation scheme based on evolutionary algorithms.

4.1 Evolution Algorithm

Evolutionary algorithm is a computational model that simulates biological evolution in nature. It can automatically accumulate the knowledge of solution space in the process of evolution and control the search process adaptively to obtain the optimal solution of the target function. EA is a mature global optimization method with high robustness and wide applicability. Therefore, we use the optimal task offloading and resource allocation scheme based on evolutionary algorithm to minimize the cost function, specifically including the following five steps: 1. Population initialization; 2. Calculate individual fitness; 3. Selection; 4. Mutation; 5. Crossover.

4.1.1 Population Initialization

First, according to the characteristics of the problem (minimization problem) and the range of variables, N population individuals are randomly generated in the solution space:

$$\left\{ X_a(0) \middle| x_{a,b}^L \le x_{a,b}(0) \le x_{a,b}^U; \mathrm{a} = 1,2,\cdots,N; \mathrm{b} = 1,2,\cdots,M \right\} \tag{19}$$

where N represents the population size, M is the dimension of the solution space. $x_{a,b}^L$ and $x_{a,b}^U$ are the upper and lower bounds of the range of values for the M dimension variables, respectively. The value of $x_{a,\mathrm{b}}(0)$ are as follow:

$$x_{a,b}(0) = x_{a,b}^L + rand(0,1) \cdot (x_{a,b}^U - x_{a,b}^L) \tag{20}$$

where $rand(0,1)$ denotes a random number, which is on the interval (0, 1).

4.1.2 Calculate Individual Fitness

The second step is to calculate the individual fitness, which can reflect the gap between the individual of each population and the optimal solution of the problem. Individuals with high fitness have a greater chance of being selected for the next evolution. The fitness of individual population can be obtained by fitness function. Considering that our objective function is a minimization problem, the fitness function Fit can be defined as follow:

$$Fit_{a,b} = \begin{cases} f_{\max} - f_{a,b}(x), f(x)_{a,b} < f_{\max} \\ 0 \qquad ,others \end{cases} \tag{21}$$

where $f_{a,b}(x)$ is the objective function, $f_{\max}$ is the maximum of $f_{a,b}(x)$.

4.1.3 Selection

After setting the fitness function, the individual population needs to be screened to leave better offspring for the next round of selection. We determine the rules of selection according to the fitness value of the individual. First, we call the sum fitness of all individuals in the population as the total fitness. Then divide the fitness of each individual by the total fitness to obtain the probability of the individual being selected. We express the selection probability as:

$$P(\mathrm{i}) = \frac{Fit(\mathrm{i})}{\sum_{i=1}^{N} Fit(\mathrm{i})} \tag{22}$$

4.1.4 Mutation

In the g-th iteration, for individual $\left\{X_a(\mathrm{g})\middle| x_{a,b}^L \le x_{a,b}(\mathrm{g}) \le x_{a,b}^U; \mathrm{a}=1,2,\cdots,N; \mathrm{b}=1,2,\cdots,\mathrm{M}\right\}$, three individuals $X_{r1}(\mathrm{g})$, $X_{r2}(\mathrm{g})$, $X_{r3}(\mathrm{g})$ are randomly selected from it, and r1≠r2≠r3. After mutation, an intermediate $V_a(\mathrm{g}+1)$ is produced:

$$V_a(\mathrm{g}+1) = X_{r1}(\mathrm{g}) + F \cdot (X_{r2}(\mathrm{g}) - X_{r3}(\mathrm{g})) \tag{23}$$

where $r1$, $r2$, $r3$ is the random number within the interval and F is the scaling factor.

4.1.5 Crossover

At the last step, the g-th generation population $\{X_a(\mathrm{g})\}$ and its mutated intermediate $\{V_a(\mathrm{g}+1)\}$ are cross-operated between individuals:

$$u_{a,b}(\mathrm{g}+1) = \begin{cases} V_{a,b}(\mathrm{g}+1), & if \quad rand(0,1) \le \omega \\ X_{a,b}(\mathrm{g}), & otherwise \end{cases} \tag{24}$$

where ω is the cross probability between two individuals.

In summary, the flow chart of the proposed method is shown in Fig. 2:

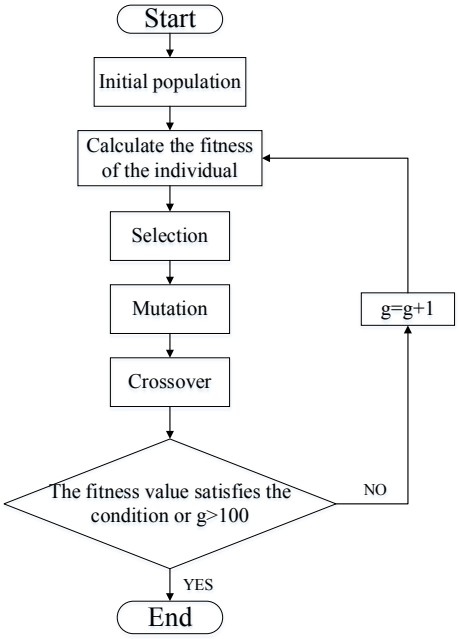

Figure 2: Flow chart of the proposed method

5 Simulation

In this section, the environmental settings of the experiment and the results of the experiment are presented to demonstrate the performance of our proposed method. First, the number of all vehicles and MEC servers in the experiment, communication resources and computing resources, and the relevant parameters are set as shown in Tab. 1:

Table 1: The parameters of experiment

Parameter	Value
Total bandwidth	1×10^6 (Hz)
Number of vehicle	9
Number of MEC server	3
Total CPU rate of MEC	4×10^{10} (cycles/s)
CPU rate of vehicle	5×10^8 (cycles/s)
Task data size	1,2,3 (MB)
Unit processing energy consumption	$1/7.3 \times 10^8$ (J/cycle)
The transmission and receiving energy consumptions of vehicle	1.42×10^{-7} (J/bit)
Tolerance delay of tasks	0.4,0.6,0.8 (s)
C_K	1900 (cycles/bit)
The received power	0.4 (W)
The noise power	0 (W)
The interference power	4 (W)

To demonstrate the function and performance of the proposed method, we introduce two other computational offloading strategies: 1. The local offloading policy (LE), i.e., the task performs all computational tasks locally; 2. MEC execution policy (ME), i.e., all tasks are offloaded to the MEC server.

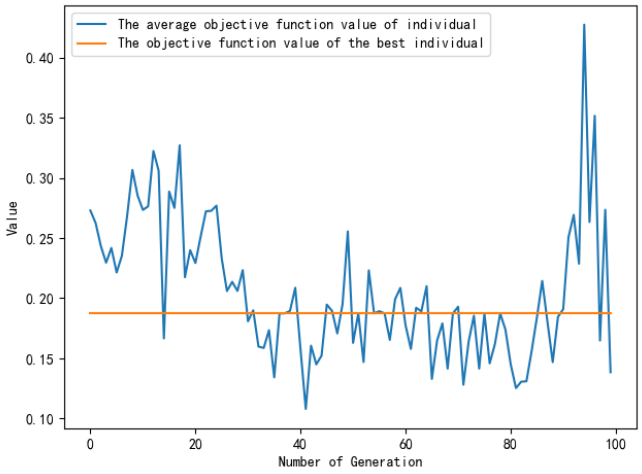

Figure 3: The value of objective function obtained by evolutionary algorithm

Table 2: The value of decision variables

Number	$x_{i,j}$	$B_{i,j}$ (Hz)	f_j^M (cycles/s)
Vehicle 1	0.24	667572	9.2×10^7
Vehicle 2	0.91	80536	4.9×10^8
Vehicle 3	0.66	63507	2.8×10^8

As shown in Fig. 3, the average objective function value of the population individual reaches the maximum at the 92nd generation, and the optimal solution of the objective function is 0.187. The values

of the decision variables are shown in Tab. 2.

We compare the proposed scheme with the ME and LE schemes, and the results are shown in Fig. 4, which shows that our method is better. Compared to the LE scheme, our scheme significantly reduces the cost of completing the user's task, which is only 13% of the cost of the LE scheme, which is manifested in the fact that the user can get the required at little time and energy. Compared with the ME scheme, the cost of Vehicle *2* and Vehicle *3* decreased slightly, by 9% and 34% respectively, while the cost of Vehicle 1 decreased by 76%. So from the overall point of view of MEC server *M*, our scheme is better than the ME scheme and can provide better QOS.

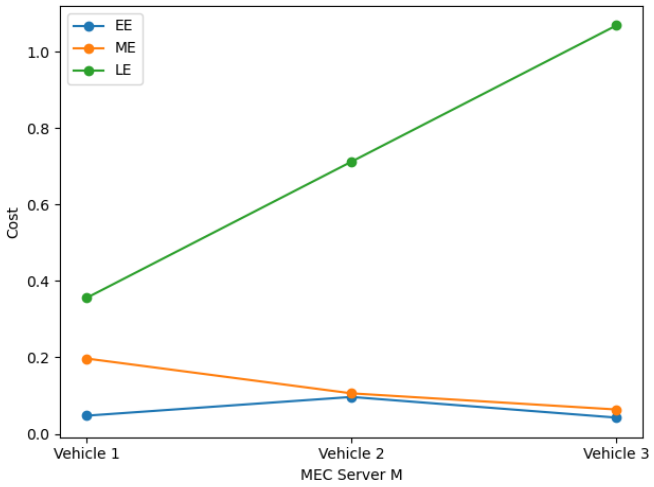

Figure 4: Performance comparison with ME and LE algorithms

5 Conclusion

In this paper, we combine the definition of vehicle edge network delay with evolutionary algorithm to improve service quality. First, all delays and energy consumption during vehicle task offloading are defined, including three processes: 1. Task offloading 2. Task computation 3. Result feedback. Then, this cost problem is modeled. Finally, the evolutionary algorithm is used to optimize the problem, including population initialization, calculating individual fitness, selection, mutation and crossover, and the optimal solution is obtained by iteration. By comparing experiments, we prove the superiority of this method, which can reduce communication delay and improve service quality (QOS).

Acknowledgement: This work was supported by the National Natural Science Foundation of China (Grant Nos. 61602252, 61802197, 61972207), the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20160967), and the Project through the Priority Academic Program Development (PAPD) of Jiangsu Higher Education Institution.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] X. Yang, X. Yu, H. Huang and H. Zhu, "Energy efficiency based joint computation offloading and resource allocation in multi-access MEC systems," *IEEE Access*, vol. 7, pp. 117054–117062, 2019.

[2] Y. Liu, J. Liu, A. Argyriou and S. Ci, "MEC-assisted panoramic VR video streaming over millimeter wave mobile networks," *IEEE Transactions on Multimedia*, vol. 21, no. 5, pp. 1302–1316, 2019.

[3] A. Sun, G. Gao, T. Ji and X. Tu, "One quantifiable security evaluation model for cloud computing platform," in *Sixth Int. Conf. on Advanced Cloud and Big Data*, pp. 197–201, 2018.

[4] M. Mehrab, D. You, V. Latzko, H. Salah, M. Reisslein and F. H. P. Fitzek, "Device-enhanced MEC: Multi-Access Edge Computing (MEC) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, pp. 166079–166108, 2019.

[5] Z. Tan, F. R. Yu, X. Li, H. Ji and V. C. M. Leung, "Virtual resource allocation for heterogeneous services in full duplex-enabled small cell networks with cache and MEC," in *IEEE Conf. on Computer Communications Workshops*, pp. 163–168, 2017

[6] J. Zhou, F, Wu, K. Zhang, Y. Mao and S. Leng, "Joint optimization of offloading and resource allocation in vehicular networks with mobile edge computing," in *10th Int. Conf. on Wireless Communications and Signal Processing*, pp. 1–6, 2018.

[7] C. You, K. Huang, H. Chae and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, 2017.

[8] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan *et al.,* "Mobility-induced service migration in mobile micro-clouds," in *IEEE Military Commun. Conf.*, pp. 835–840, 2014.

[9] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[10] Q. Liu, T. Han, N. Ansari, "Joint radio and computation resource management for low latency mobile edge computing," in *2018 IEEE Global Communications Conf.*, pp. 1–7, 2018.

[11] C. Luo, S. Salinas, M. Li and P. Li, "Energy-efficient autonomic offloading in mobile edge computing," *IEEE 15th Int. Conf. on Dependable, Autonomic and Secure Computing*, pp. 581–588, 2017.

[12] M. Xiao, N. B. Shroff and E. K. P. Chong, "A utility-based power-control scheme in wireless cellular systems," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 210–221, 2003.

[13] J. Wang, W. Wu, Z. Liao, A. K. Sangaiah and R. Simon Sherratt, "An energy-efficient off-loading scheme for low latency in collaborative edge computing," *IEEE Access*, vol. 7, pp. 149182–149190, 2019.

[14] K. Kim, J. Lynskey, S. Kang, C. S. Hong, "Prediction based sub-task offloading in mobile edge computing," in *2019 Int. Conf. on Information Networking*, pp. 448–452, 2019.

[15] Q. Liu, T. Han and N. Ansari, "Joint radio and computation resource management for low latency mobile edge computing," in *2018 IEEE Global Communications Conference*, pp. 1–7, 2018.

[16] Y. Z. Song, S. Yau, R. Z. Yu, X. Zhang and G. L. Xue, "An approach to QoS-based task distribution in edge computing networks for IoT applications," in *2017 IEEE International Conference on Edge Computing,* pp. 32–39, 2017.

[17] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas and S. Khaf, "A deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633, 2019.

[18] K. Zhang, X. Gui and D. Ren, "Joint optimization on computation offloading and resource allocation in mobile edge computing," in *2019 IEEE Wireless Communications and Networking Conf.*, pp. 1–6, 2019.

[19] H. Chen, K. Liu, C. Ma, Y. Han and J. Su, "A novel time-aware frame adjustment strategy for RFID anti-collision," *Computers, Materials & Continua*, vol. 57, no. 2, pp. 195–204, 2018.