Cheng and Guan *J Wireless Com Network* (2021) 2021:101
https://doi.org/10.1186/s13638-021-01984-6

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Research on task-offloading decision mechanism in mobile edge computing-based Internet of Vehicle

Jun Cheng[1] and Dejun Guan[2*]

*Correspondence: sunauthor@126.com
[2] First Affiliated Branch, Shenyang Radio and TV University Shenyang, Liaoning 110003, China
Full list of author information is available at the end of the article

Abstract

As a technology integrated with Internet of things, mobile edge computing (MEC) can provide real-time and low-latency services to the underlying network and improve the storage and computation ability of the networks instead of central cloud infrastructure. In mobile edge computing-based Internet of Vehicle (MEC-IoV), the vehicle users can deliver their tasks to the associated MEC servers based on offloading policy, which improves the resource utilization and computation performance greatly. However, how to evaluate the impact of uncertain interconnection between the vehicle users and MEC servers on offloading decision-making and avoid serious degradation of the offloading efficiency are important problems to be solved. In this paper, a task-offloading decision mechanism with particle swarm optimization for MEC-IoV is proposed. First, a mathematical model to calculate the computation offloading cost for cloud-edge computing system is defined. Then, the particle swarm optimization is applied to convert the offloading of task into the process and obtain the optimal offloading strategy. Furthermore, to avoid falling into local optimization, the inertia weight factor is designed to change adaptively with the value of the objective function. The experimental results show that the proposed offloading strategy can effectively reduce the energy consumption of terminal devices while guarantee the service quality of users.

Keywords: Task-offloading, Particle swarm optimization, Internet of things, Mobile edge computing, Internet of Vehicle

1 Introduction

With the rapid development of Internet of things, the integration of the storage and processing capabilities of plenty of Internet of things (IoT) devices should be inevitable for providing real-time and low-latency services [1]. In recent years, massive data computing and high-quality customer experience services such as unmanned driving, automatic navigation, ultra high definition video, virtual reality, augmented reality, online games have gradually been emerged. Due to long communication delay and high operation cost, remote computing task loading mode based on cloud computing technology will face severe challenges. Meanwhile, with the impressive growth of mobile terminals, it is urgent to solve the contradiction among the

limited computing power, long-term continuous low latency and high quality of service requirements [2]. As a technology usually integrated with IoT, MEC can supply services and cloud computing functions needed by wireless users nearby and has the potential to provide real-time, low-latency services to the underlying network [3]. Compared with the traditional cloud architecture, the number of data centers with mobile edge computing is relatively small, and seems to be more suitable for the delay, responsiveness and privacy requirements of those advanced services. At present, it is a trend to promote the development of MEC technology through the integration of technology of mobile computing and wireless communication [4–6]. In order to further reduce the response delay of user requests and the energy consumption of mobile terminal and network, it is necessary to exploit more feasible ways to improve the efficiency of resource utilization.

In order to improve the urban traffic conditions, the Internet of Vehicle (IoV) as a new paradigm is introduced to enhance the information interaction between vehicles and people [7]. In the IoV environment, the vehicle is connected to devices such as smart cameras, sensors and actuators. By the transmitters and receivers in above signal collecting system, vehicles can connect to remote infrastructure and other vehicles [8]. However, with the rapid increase in road traffic, the gap between the requirements of communication services and the limited vehicle capacity has become a serious problem. The on-board network is facing the challenge of providing ubiquitous connection and high-quality service for many vehicles. In order to solve those problems, MEC has been explored as a promising technology in vehicular networks by using computing resources at the edge of vehicular wireless access networks [9, 10]. As a typical application of IoT technology in the field of the intelligent transportation system, IoV can realize intelligent traffic management and introduce more mature conventional applications, which include path planning, navigation, autonomous driving, intelligent-assisted driving and augmented reality for vehicles, online interactive games and other rich media applications for passengers [11, 12]. However, focusing on the various computing tasks with different granularity and quality of service (QoS) requirements, it is not enough to rely on the lightweight edge server placed on the roadside unit. It is a great challenge to ensure the normal and efficient operation of those highly complex services, especially to provide stable and reliable connections and high-quality network services for a large number of vehicles [13, 14].

MEC can be applied as the core access points for task transmission and processing of the edge computing resources in IoV [15]. Due to the delay caused by the distance between the edge IoT devices and the data center, the function related to the average task latency and resource cost should not be ignored in the centralized cloud model [16, 17]. As the number of edge devices grows exponentially, high latency can be a huge challenge for many applications that involve end-to-end communication.

The remainder of the paper is organized as follows. Section 2 reviews the related works. Section 3 presents problem formulation. The proposed task-offloading decision mechanism is presented in Sect. 4. The simulations and analysis are provided in Sect. 5, and the conclusions are presented in Sect. 6.

2 Related work

Due to the limited computing power, storage and energy of mobile devices, task scheduling is a classic method to transfer tasks to external platforms [18, 19]. It can improve computing efficiency, reduce task completion time and effectively make use of the resources of other devices. Since the edge computing network has the characteristics of ultra-dense deployment and simultaneous access of a large number of users, the selection of user computing task mode is extremely important and directly determines the computing time and cost of the system.

Wang et al. [20] designed a multilayer model integrated with user layer, access layer and cloud to jointly process user data, which defined the allocated computing and transmission resources for each device and constructed a convex optimization problem to obtain the optimal resource allocation strategy. Zhang et al. [21] exploited the virtualization technology to allocate resources online in dense cloud wireless access network and used Lyapunov optimization theory to achieve the tradeoff between the average energy consumption and delay. Prof. Chen et al. [22] proposed an improved the search tree algorithm by using the branch and bound method to solve the delay minimization problem of computational offloading and resource allocation. By establishing the cloud-edge cooperation model, Zhao et al. [23] designed the optimal decision making scheme to transfer the request to the edge server or cloud server for serial processing, in which different mobile device requests should pass through the access point in chronological order. In order to maximize resource utilization, Dr. Ning et al. [24] proposed to combine multiple edge servers for cache allocation and computational offload. Based on the Stackelberg game theory, Salahuddin et al. [25] introduced a strategy to decide whether to unload the computation tasks or not according to the decision of the operator and other devices, so as to achieve the optimal profit. Lee et al. [26] adopted the idea of user centered fair resource allocation and defined a user-centered utility function with weighted the user's delay tolerance, bit error rate and energy consumption.

The introduction of edge computing can greatly shorten the physical distance between on-board tasks and computing resources. Compared with the central cloud infrastructure, the IoV cloud-edge computing can provide real-time and low-latency services. Aiming at the problem of energy consumption control based on edge server, Kumar et al. [27] constructed an efficient and energy-saving IoV resource scheduling framework based on mobile edge computing for large scale and wide distribution of the vehicles. Yu et al. [28] proposed a MEC based offloading method in IoV for the selection of the optimal MEC server for task management. The method comprehensively considered the vehicle mobility and computing tasks to make the offloading decision. Zhang et al. [29] developed an energy-saving computing offloading scheme in multi-user fog computing system and proposed a distributed algorithm based on the alternating direction multiplier method. Wang et al. [30] designed a framework for jointly optimizing task allocation decisions and the CPU frequency, in which semidefinite relaxation algorithms are proposed to solve the problem of fixed and elastic CPU frequency. Zuo et al. [31] investigated the power minimization problem under the constraint of task buffer stability and proposed an online algorithm based on Lyapunov optimization for local execution and computational offloading. Ma et al. [32] investigated the multi-user computational offloading problem in multi-channel wireless interference environment and proposed a

game theory computational offload algorithm. Xiong et al. [33] designed an optimization algorithm for the allocation of network resources and computing resources to minimize the transmission delay and computation time. Aiming to minimize the average response time of vehicle computing tasks, Dai et al. [34] implemented the real-time traffic management of vehicle offloading based on fog computation. The vehicle based fog node is modeled mathematically according to the queuing theory, and an approximate method for solving the unloading optimization problem is introduced.

The cost for using the cloud resources is neglected in above methods. Hence, in this paper, we will provide a task-offloading decision mechanism with particle swarm optimization, which considers both the completion time of tasks and monetary cost of computing resources.

3 Problem formulation

Consider a IoV cloud-edge computing (IoV-CEC) system with N vehicle users and multiple MEC servers to perform computation offloading operation. Assuming that there is a bidirectional road and all computing devices share the same uplink spectrum of the system. All channel gains are modeled as large-scale path loss and small-scale Rayleigh fading, and H_{ij} denotes the channel gain from ith vehicular user to associated jth MEC server. The additional noise of the user's destination is an independent cyclosymmetric complex Gaussian random variable with mean zero and variance σ^2. E_i denotes the transmission power of the ith user is, and C_i denotes the number of CPU cycles required by the user to execute unit data. Assume that the total amount of workload for ith user is L_i, αL_i will be offloaded to the MEC and the rest will be processed locally. The user's local CPU frequency is F_i, which is measured by the number of CPU cycles per second. Then, the time consumption for local computing of the ith user can be given by:

$$T_{\text{Local},i} = \frac{(1-\alpha)L_i C_i}{F_i} \tag{1}$$

Once the IoV user conducts the computation offloading decision, the time consumption of the offloading task includes the time consumption of workload transmission in uplink channel, the time consumption of task execution on the MEC server and the time consumption of results feedback. Therefore, the time consumption of computation offloading can be calculated by:

$$T_{\text{OFF},i} = t_{u,i} + t_{c,i} + t_{d,i} \tag{2}$$

where $t_{u,i}$, $t_{c,i}$ and $t_{d,i}$ represent the time consumption of workload transmission in uplink channel, task execution on the MEC server and results feedback, respectively.

Since the local processing and offloading of the workload can be performed simultaneously, the time consumption of ith user to execute the task can be expressed as follows:

$$T_i = \max\{T_{\text{Local},i}, T_{\text{OFF},i}\} \tag{3}$$

After the offloaded task is processed, the corresponding results will be sent back to the IoV user through the downlink channel. Let βL_k denote the amount of the results, and β represents the ratio of the output to input offloaded workload. Hence, the time consumption of result transmission through downlink channel can be given by:

$$t_{d,i} = \frac{\beta L_i}{\upsilon_i} \tag{4}$$

where υ_i is the uplink transmission rate of ith user. According to Shannon's theory, the uplink transmission rate can be defined as:

$$\upsilon_i = B_i \log_2 \left(1 + \frac{E_i H_i}{B_i N_0}\right) \tag{5}$$

where N_0 is the power spectral density of the noise. Besides, B_i is the communication band obtained by the communication between the user and the MEC server, i.e., the bandwidth being occupied by ith user for offloaded workload.

Furthermore, if f_i denotes the computing ability allocated to ith user by the associated MEC server, the time consumption of task processing can be obtained by

$$t_{c,i} = \frac{\alpha L_i C_i}{f_i} \tag{6}$$

For simplicity, the equal distribution is taken into account, and we have

$$f_i = \frac{F_c}{N} \tag{7}$$

where F_c represents the total computing ability of the MEC server and can be measured by the number of CPU cycles per second.

Meanwhile, owing to the user's communication overhead of the interaction with the MEC server, the user's transmission power can be regarded as a constraint condition, and it can be described by $E_i \leq E_{\max}$.

Since the user will consume the resources of the MEC server to perform the offloading task, the MEC server should provide enough computing capacity to complete the offloaded task. Therefore, in order to balance the demand and supply of resources, it is necessary to consider the energy consumption of the data generated by the communication between vehicle users and the MEC servers when they make offloading decisions.

The energy consumption for local computing can be given by

$$E_{\text{Local},i} = \gamma(1-\alpha)L_i \tag{8}$$

where γ is the energy consumption of the user's CPU.

The MEC server can rent the corresponding computing resources to the user, and the user will cause energy consumption for workload transmission during the course of offloading, which can be expressed by:

$$E_{\text{Trans},i} = \frac{E_i \alpha L_i}{B_i \log_2 \left(1 + \frac{E_i H_i}{B_i N_0}\right)} \tag{9}$$

Similarly, the energy consumption of the computing tasks offloaded from ith user in the MEC server can be given by

$$E_{\text{OFF},i} = \gamma_c \alpha L_i \tag{10}$$

where γ_c is the CPU's energy consumption of the MEC server.

Let $\mu = \{\mu_1, \mu_2, \cdots, \mu_N\}$ be the monetary cost corresponding to the user's energy consumption and the goal of IoV-MEC system is to maximize the utility of MEC provider by offering limited resources to IoV users. Mathematically, the optimization problem can be formulated by

$$\mathbf{P1}: \quad \max_{\mu} \ U_c(\mu) = \sum_{i=1}^{N} \mu_i E_{\mathrm{OFF},i} \tag{11}$$

For IoV user, the cost of the workload includes the energy consumption of local processing with specific delay, the energy consumption for communication, and the energy consumption of task processing at the MEC server, which can be calculated by

$$C_i(L_i, \mu_i) = E_{\mathrm{Local},i} + E_{\mathrm{Trans},i} + \mu_i E_{\mathrm{OFF},i} \tag{12}$$

For each IoV user, the determination of the optimal offloading workload according to the monetary cost of the MEC servers can reduce the payment to the greatest extent. Therefore, the optimization problem can be described as follows:

$$\mathbf{P2}: \quad \min \sum_{i=1}^{N} C_i(L_i, \mu_i) \tag{13}$$

$$s.t. \quad E_i \le E_{\max}$$

$$\sum_{i=1}^{N} \alpha L_i C_i \le F_{\max}$$

where $F_{\max}$ is the maximum processing time of MEC server.

4 Proposed method

Particle swarm optimization is a swarm intelligence optimization algorithm, which is derived from the behavior of birds looking for habitat by Kennedy and Eberhart in 1995. It has the advantages of less parameters, easy implementation and fast convergence [35]. For computing-intensive task offloading, the process can be regarded as different task particles to choose their best location and to quickly search for the location set in accordance with the scene under different conditions [36]. Therefore, the particle swarm optimization (PSO) method is adopted to realize the transformation between the decision-making of unloading and the dynamic search process. In PSO, according to the foraging behavior of birds, the search space of the problem is compared with the flight space of birds. Each bird is treated as a particle to represent the search for the target food, and the position of the particles is evaluated by the defined fitness function [37]. During the search phrase, the best position found can be also shared and dynamically the position of particles should be adjusted through their own experience and peer experience.

Suppose the ith user generates $M_i\left(B_i, f_i, E_{\max}\right)$ tasks, and $S = \{s_1, s_2, \cdots, s_K\}$ denotes the set of all MEC servers, and the maximum transmission rate of tasks transmitted to MEC server can be calculated by the matrix of channel gain:

$$H = \begin{bmatrix} 0 & H_{1,2} & \cdots & H_{1K} \\ H_{21} & 0 & \cdots & H_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ H_{N1} & H_{N2} & \cdots & H_{NK} \end{bmatrix} \tag{14}$$

where H_{ij} is the channel gain for the task generated by ith user to be transmitted to jth MEC server.

The particles will represent a specific MEC server to which all tasks will be offloaded, and the dimension of particle is the same as the number of task sets [38]. The offloading decision vector of each particle is $X[i]$, which can represent the optimal execution location of all tasks and each element is randomly selected from 0 to K. $X[i] = 0$ indicates that the current task is executed locally. Otherwise, it indicates that the current task is loaded to the corresponding MEC server.

The speed of particles indicates the processing efficiency of the current task assigned to other MEC servers, which is recorded as $V[i]$ [39, 40]. The dimension of particle speed is the same as the number of task sets. The optimal position of each particle during all iterations is denoted as Pbest, and the optimal position of all particles is denoted as Gbest.

To reflect the total cost of all tasks assigned to different MEC servers, the fitness of particles can be represented by the combination of above optimization problems. Thus, the fitness function can be given by

$$g(\mu_i) = \sum_{i=1}^{N} C_i(L_i, \mu_i) / \sum \mu_i E_{\mathrm{OFF},i} \tag{15}$$

The update of the particle's velocity can be given by:

$$V[i] = w_i V[i] + c_1 \mathrm{rand}\big(\mathrm{pbest}_i - X[i]\big) + c_2 \mathrm{rand}\big(\mathrm{gbest}_i - X[i]\big) \tag{16}$$

where c_1, c_2 are the learning factors, and rand is a random number between 0 and 1, and w_i is the inertia weight factor.

To avoid falling into local optimization, the inertia weight factor should be dynamically changed with the objective function value of particle swarm optimization. Hence, the inertia weight factor w_i will be updated according to the following equation:

$$w_i = \begin{cases} w_{\max}, & g_i > g_{\max} \\ w_{\max} - \frac{(w_{\max} - w_{\min})(g_{\max} - g_i)}{g_{\max} - g_{\min}}, & g_{\min} \le g_i \le g_{\max} \\ w_{\min}, & g_i < g_{\min} \end{cases} \tag{17}$$

where $w_{\min}$ and $w_{\max}$ represent the minimum and maximum value of the inertia weight factor, respectively. In addition, g_i is the current fitness value of each particle. $g_{\min}$ and $g_{\max}$ are the average fitness value of all particles whose fitness value is lower or greater than the average fitness value of the particle swarm, respectively.

Furthermore, the position of the particles can be updated as follows:

$$X[i] = X[i] + V[i] \tag{18}$$

Next, the detailed computation offloading process is demonstrated as follows:

Step 1: Initialize the random position and velocity of ith particle. The delay and energy consumption are measured by local tasks, MEC server set and channel gain matrix H. The fitness of each particle is calculated according to Eq. (15).

Step 2: The current position of the particles is chosen as the individual optimal allocation scheme Pbest, and the position of the particle with the minimum fitness value will be set as Gbest.

Step 3: If $t \leq t_{\max}$, the velocity $X[i]$ and the position $V[i]$ will be updated independently according to all dimensions for ith particle.

Step 4: The updated position of each particle is calculated according to Eq. (17). If the corresponding fitness value is less than the current fitness value, the particle optimal Pbest will be replaced. Meanwhile, the population optimal allocation scheme Gbest and the total cost fitness will be updated.

Step 5: At the end of iterations, the optimal vector $X[i] =$ Gbest and the offloading strategy with optimal fitness can are obtained. Through centralized control, the cloud-edge computing enables the IoV users to offload the computing tasks to MEC servers for execution according to the optimal offloading strategy.

5 Results and discussion

In this section, the simulation experiments are performed to evaluate the performance of the proposed algorithm. An MEC-IoV system with a hexagonal symmetric cellular structure is considered, in which the number of MEC servers is 6 and the number of IoV users is 10. The reference signal-to-noise ratio is set to be 20 dB, and the bandwidth of uplink and that of downlink is 20 MHz. The computing ability of the MEC server is 4 GHz, the CPU cycles required for computing one bit of the task are between 500 and 3000, and the maximum latency for completing the task is set to 5 s. The channel gain for the task generated by the IoV user to be transmitted to associated MEC server is varied from $[2 \times 10^{-10}, 2 \times 10^{-6}]$. In addition, the other parameters is set as following: $E_i = 0.4$ W, $f_i = 500$ MHz, $\alpha = 0.3, \gamma = 2 \times 10^{-8}$ J/bit, $c_1 = c_2 = 2$, respectively.

First, the convergence of the proposed unloading strategy is evaluated. The total cost of all tasks can be obtained with the number of iterations, and the experimental results are shown in Fig. 1. It can be observed that our proposed algorithm converges quickly in the early iterations. After 30 iterations, the total cost remains relatively stable and the global optimal solution can be found. Our proposed algorithm has a strong ability of global optimization. It can constantly seek the global optimal solution in the early stage and also has good global search ability in the later stage. Compared with greedy algorithm, the total cost of our proposed algorithm is much less and reduces the total cost by 14.2%.

In addition, the utility of MEC provider under different monetary cost is analyzed as shown in Fig. 2. From the experimental results, we can see that when the monetary cost changes between [1, 2.6], the utilities of MEC provider show a linear monotonic increasing trend. However, when the monetary cost exceeds 2.6, the utility of MEC provider begins to decline sharply, and tends to be relatively stable at around 1.65. The reason is that the user's offload costs increase as well as the monetary cost. Compared with the benefits of task offloading, the user's spending exceeds the corresponding cost, which results in the user unwilling to make offloading decision. The greedy algorithm also

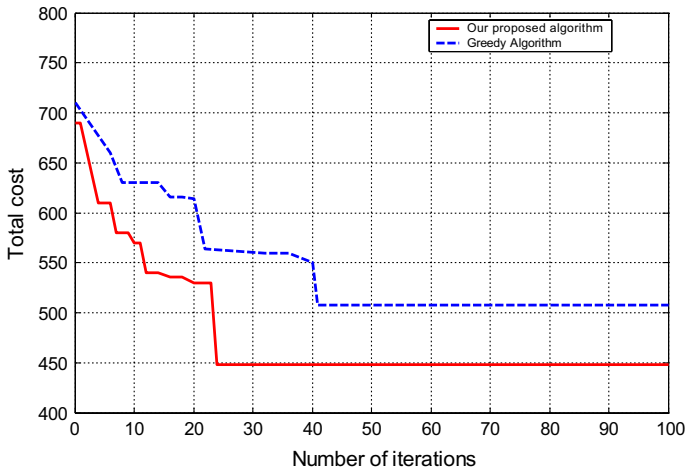

Fig. 1 Total cost versus number of iterations

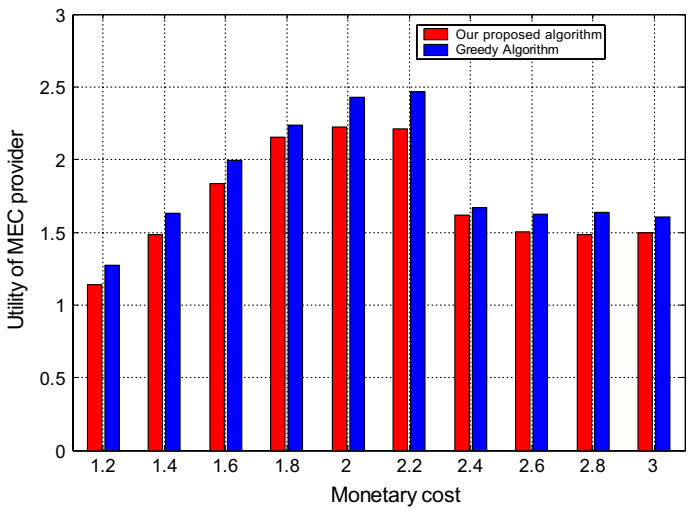

Fig. 2 Utility of MEC provider versus monetary cost

shows a similar trend, but on the whole, the utility of MEC provider is lower than our proposed algorithm.

Next, we explore the performance in aspect of average resource cost, utility of MEC provider, computation overhead and user's offloading ratio and conduct the comparison with random scheme and round-robin scheme. Among them, random offload scheme refers to the random offloading of tasks to the MEC server for processing, while the round-robin scheme is to offload the tasks to the MEC server in sequence. Figure 3 shows the curve of the average resource cost with different amount of workload. It can be seen that the average resource cost is not conducive to the workload. In comparison, the average resource cost shows good stability in our proposed algorithm and fluctuates sharply in Round-Robin scheme and Random scheme. With the increase of the amount of workload, the monetary cost will gradually decline. It will

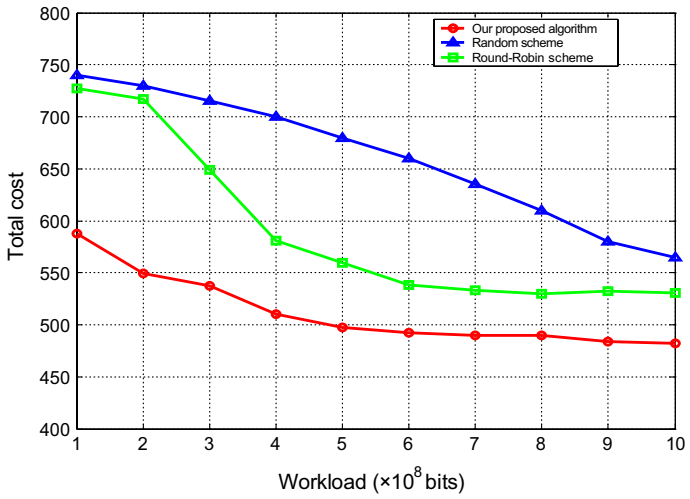

Fig. 3 Average resource cost versus workload

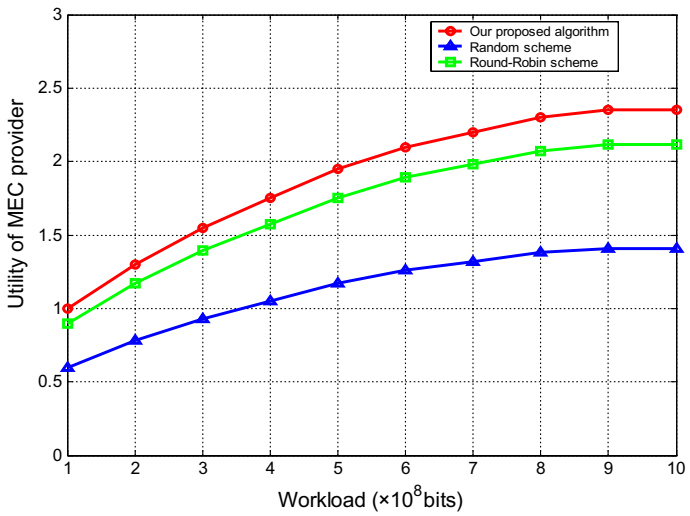

Fig. 4 Utility of MEC provider versus workload

cause that the user's own processing capacity cannot meet the corresponding requirements. Moreover, with the increase of the amount of offloading, the MEC server can utilize the low cost to encourage users for more offloading tasks, which will bring more benefits to the system. When the user's workload reaches about 5×10^8, the curve of average resource cost begins to flatten. The reason is that the users have to offload the task within the requirement as much as they will process.

Figure 4 shows the curve of the utility of MEC provider with different amount of workload. The simulation results show that with the increase of user data, the utility of MEC provider of our proposed algorithm, round-robin scheme and random scheme algorithms is on the rise. Also, it is obvious that the utility in random scheme algorithm is much lower than that of our proposed algorithm and round-robin scheme.

At high level of workload, the utility of MEC provider in our proposed algorithm and Round-Robin scheme increases rapidly. Overall, the performance of our proposed algorithm is better than that of other schemes. It is clearly evident from Fig. 4 that the utility of the system gradually tends to be flat when the workload exceeds 8×10^8. That is because with the increase of users the processing capacity has reached saturation, and the utility of MEC provider will not fluctuate at high level of workload.

Figure 5 shows the comparison of computation overhead. With the increase of workload, the computation overhead of all algorithms shows linear growth trend. In our proposed algorithm, the particle swarm optimization can obtain an optimal offloading decision and effectively reduce the computation overhead. The results also show that the increase of computation overhead in random scheme is obviously greater than that of our proposed algorithm and round-robin scheme. The low monetary cost will lead to large-scale offloading of users. However, the communication cost of users and the computation overhead increase dramatically, which directly leads to the sharp increase of overall cost.

Figure 6 shows the curve of the user's offloading ratio with the amount of workload. When the amount of workload is very low, the cost of local processing and offloading to MEC server is almost equivalent. Thus, the IoV users tend to offload tasks to the MEC server for execution. However, with the increase of user data, the difference of the cost between local computing and offloading decision becomes distinctly. Especially, when the amount of workload increases to a certain extent, the revenue of MEC server will no longer fluctuate and the user's offloading ratio begins to decline. From the experimental results, we can observe that when the amount of workload is 5×10^8, the user's offload ratio of round-robin scheme shows a downward trend. In this case, the computation overhead starts to increase sharply, and it is more cost-effective for users to choose local processing, which leads to the decrease of offloading ratio. In our proposed algorithm, the user's offloading ratio begins to fluctuate as the workload exceed 5×10^8, and demonstrates better than other schemes.

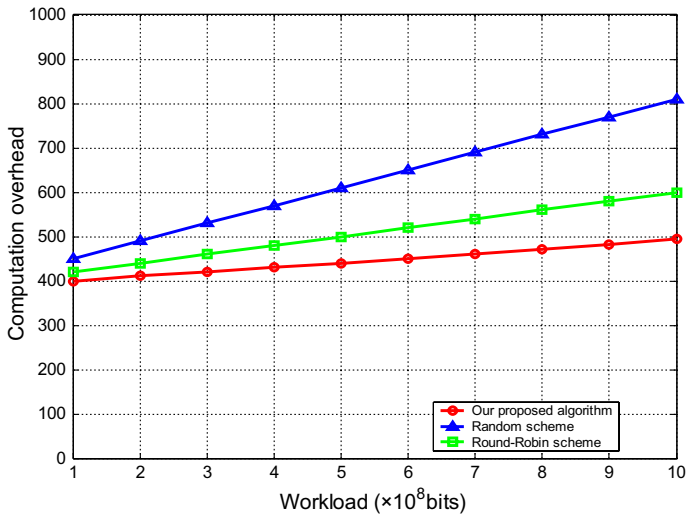

Fig. 5 Computation overhead versus workload

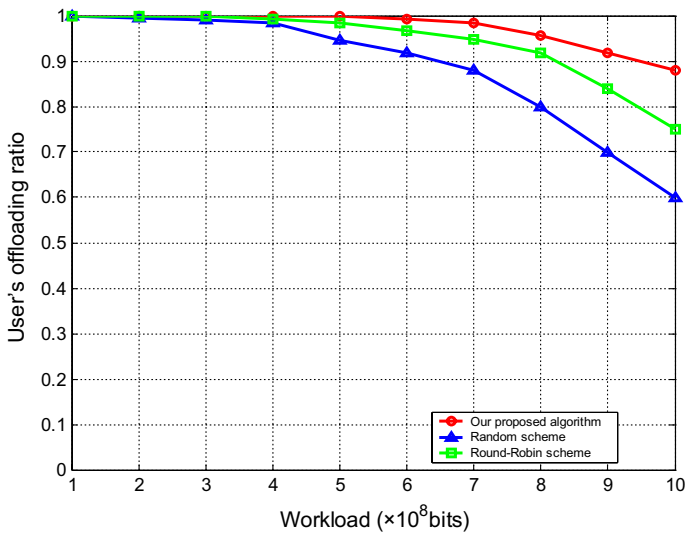

Fig. 6 User's offloading ratio versus workload

6 Conclusions

In this paper, we proposed a task-offloading decision mechanism with particle swarm optimization for IoV-based edge computing. First, a mathematical model to calculate the computation offloading cost for cloud-edge computing system is defined. Then, the particle swarm optimization is applied to convert the offloading of task into the process and obtain the optimal offloading strategy. Furthermore, to avoid falling into local optimization, the inertia weight factor is designed to change adaptively with the value of the objective function. The experimental results show that the proposed offloading strategy can effectively reduce the energy consumption of terminal devices while guarantee the service quality of users. In the follow-up research process, I will consider classifying tasks, such as QoS constraints, CPU requirements and priority.

Authors' contributions
JC designed research, performed research, analyzed data and wrote the paper. DG analyzed the data and was involved in writing the manuscript. All authors read and approved the final manuscript.

Funding
This work was supported by Anhui Key Research and Development Plan (Grant No. 201904a05020091), and Key scientific research projects of Chaohu University (Grant No. xlz-201905).

Availability of data and materials
Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

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

Author details
[1] College of Information Engineering, Chaohu Univeristy, Chaohu 238000, China. [2] First Affiliated Branch, Shenyang Radio and TV University Shenyang, Liaoning 110003, China.

Received: 9 February 2021 Accepted: 13 April 2021
Published online: 21 April 2021

References

1. M. Amadeo, C. Campolo, A. Molinaro, Information-centric networking for connected vehicles: a survey and future perspectives. IEEE Commun. Mag. **54**(2), 98–104 (2016)
2. S. Xu, J. Wang, W. Shou, T. Ngo, A.M. Sadick, X. Wang, Computer vision techniques in construction: a critical review. Arch. Comput. Methods Eng. (2020). https://doi.org/10.1007/s11831-020-09504-3
3. R.W.L. Coutinho, A. Boukerche, A.A.F. Loureiro, Design guidelines for information-centric connected and autonomous vehicles. IEEE Commun. Mag. **56**(10), 85–91 (2018)
4. S. He, F. Guo, Q. Zou, H. Ding, MRMD2.0: a python tool for machine learning with feature ranking and reduction. Curr. Bioinform. **15**(10), 1213–1221 (2020)
5. Q. Jiang, F. Shao, W. Gao, Z. Chen, G. Jiang, Y.S. Ho, Unified no-reference quality assessment of singly and multiply distorted stereoscopic images. IEEE Trans. Image Process. **28**(4), 1866–1881 (2018)
6. B. Wang, B.F. Zhang, F.C. Zou, Y. Xia, A kind of improved quantum key distribution scheme. Optik **235**(6), 166628 (2021)
7. T. Ni, H. Chang, T. Song, Q. Xu, Z. Huang, H. Liang, A. Yan, X. Wen, Non-intrusive online distributed pulse shrinking-based interconnect testing in 2.5 D IC. IEEE Trans. Circuits Syst. II Express Briefs **67**(11), 2657–2661 (2019)
8. X. Xue, K. Zhang, K.C. Tan, L. Feng, J. Wang, G. Chen, X. Zhao, L. Zhang, J. Yao, Affine transformation-enhanced multifactorial optimization for heterogeneous problems. IEEE Trans. Cybern. (2020). https://doi.org/10.1109/TCYB.2020.3036393
9. L. Ding, S. Li, H. Gao, Y. Liu, L. Huang, Z. Deng, Adaptive neural network-based finite-time online optimal tracking control of the nonlinear system with dead zone. IEEE Trans. Cybern. **51**(1), 382–392 (2021)
10. T. Ni, Y. Yao, H. Chang, L. Lu, H. Liang, A. Yan, Z. Huang, X. Wen, LCHR-TSV: Novel low cost and highly repairable honeycomb-based TSV redundancy architecture for clustered faults. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **39**(10), 2938–2951 (2019)
11. S. Yang, T. Gao, J. Wang, B. Deng, B. Lansdell, B. Linares-Barranco, Efficient spike-driven learning with dendritic event-based processing. Front. Neurosci. **15**, 97 (2021). https://doi.org/10.3389/fnins.2021.601109
12. A. Ndikumana, S. Ullah, T. LeAnh, N.H.Tran, Collaborative cache allocation and computation offloading in mobile edge computing, in: *Proceedings of 2017 19th Asia–Pacific network operations and management symposium (APNOMS)*, pp. 366–369 (2017)
13. H. Xu, C. Zhang, H. Li, P. Tan, J. Ou, F. Zhou, Active mass driver control system for suppressing wind-induced vibration of the Canton Tower. Smart Struct. Syst. **13**(2), 281–303 (2014)
14. K. Shi, Y. Tang, X. Liu, S. Zhong, Secondary delay-partition approach on robust performance analysis for uncertain time-varying Lurie nonlinear control system. Optim. Control Appl. Methods **38**(6), 1208–1226 (2017)
15. Q. Jiang, F. Shao, W. Lin, K. Gu, G. Jiang, H. Sun, Optimizing multistage discriminative dictionaries for blind image quality assessment. IEEE Trans. Multimedia **20**(8), 2035–2048 (2018)
16. B. Bai, Z. Guo, C. Zhou, W. Zhang, J. Zhang, Application of adaptive reliability importance sampling-based extended domain PSO on single mode failure in reliability engineering. Inf. Sci. **546**(2), 42–59 (2021)
17. S. Yang, J. Wang, X. Hao, H. Li, X. Wei, B. Deng, K.A. Loparo, BiCoSS: toward large-scale cognition brain with multi-granular neuromorphic architecture. IEEE Trans. Neural Netw. Learn. Syst. (2021). https://doi.org/10.1109/TNNLS.2020.3045492
18. S. Lv, Y. Liu, PLVA: privacy-preserving and lightweight V2I authentication protocol. IEEE Trans. Intell. Transp. Syst. (2021). https://doi.org/10.1109/TITS.2021.3059638
19. Z. Liu, L. Lang, B. Hu, L. Shi, B. Hunag, Y. Zhao, Emission reduction decision of agricultural supply chain considering carbon tax and investment cooperation. J. Clean. Prod. **294**(4), 126305 (2021)
20. P. Wang, C. Yao, Z. Zheng, Joint task assignment, transmission and computing resource allocation in multilayer mobile edge computing systems. IEEE Internet Things J. **6**(2), 2872–2884 (2018)
21. Q. Zhang, L. Gui, F. Hou, Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN. IEEE Internet Things J. **7**(4), 3282–3299 (2020)
22. Y. Chen, W. Zheng, W. Li, Y. Huang, Large group activity security risk assessment and risk early warning based on random forest algorithm. Pattern Recogn. Lett. **144**(4), 1–5 (2021)
23. J. Zhao, J. Liu, J. Jiang, F. Gao, Efficient deployment with geometric analysis for mmWave UAV communications. IEEE Wirel. Commun. Lett. **9**(7), 1115–1119 (2020)
24. Z. Ning, J. Huang, X. Wang, Mobile edge computing-enabled internet of vehicles: toward energy-efficient scheduling. IEEE Netw. **33**(5), 198–205 (2019)
25. M.A. Salahuddin, A. Al-Fuqaha, M. Guizani, Software-defined networking for RSU clouds in support of the Internet of Vehicles. IEEE Internet Things J. **2**(2), 133–144 (2015)
26. E.-K. Lee, M. Gerla, G. Pau, U. Lee, J.-H. Lim, Internet of vehicles: from intelligent grid to autonomous cars and vehicular fogs. Int. J. Distrib. Sensor Netw. **12**(9), 1–14 (2016)
27. N. Kumar, S. Zeadally, J.J. Rodrigues, Vehicular delay-tolerant networks for smart grid data management using mobile edge computing. IEEE Commun. Mag. **54**(10), 60–71 (2016)
28. C. Yu, B. Lin, P. Guo, Deployment and dimensioning of fog computing-based internet of vehicle infrastructure for autonomous driving. IEEE Internet Things J. **6**(1), 149–160 (2018)
29. K. Zhang, J. Zhang, X. Ta, C. Yao, L. Zhang, Y. Yang, J. Wang, J. Yao, H. Zhao, History matching of naturally fractured reservoirs using a deep sparse autoencoder. SPE J. (2021). https://doi.org/10.2118/205340-PA
30. X. Wang, Z. Ning, L. Wang, Offloading in Internet of vehicles: a fog-enabled real-time traffic management system. IEEE Trans. Ind. Inform. **14**(10), 4568–4578 (2018)

31. C. Zuo, Q. Chen, L. Tian, L. Waller, A. Asundi, Transport of intensity phase retrieval and computational imaging for partially coherent fields: the phase space perspective. Opt. Lasers Eng. **71**(8), 20–32 (2015)
32. X. Ma, K. Zhang, L. Zhang, C. Yao, J. Yao, H. Wang, W. Jian, Y. Yan, Data-driven niching differential evolution with adaptive parameters control for history matching and uncertainty quantification. SPE J. **26**, 993–1010 (2021)
33. Z. Xiong, N. Xiao, F. Xu, X. Zhang, Q. Xu, K. Zhang, C. Ye, An equivalent exchange based data forwarding incentive scheme for socially aware networks. J. Signal Process. Syst. **93**(11), 1–15 (2020)
34. Y. Dai, D. Xu, S. Maharjan, Y. Zhang, Joint load balancing and offloading in vehicular edge computing and networks. IEEE Internet Things J. **6**(3), 4377–4387 (2019)
35. C. Zuo, J. Sun, J. Li, J. Zhang, A. Asundi, Q. Chen, High-resolution transport-of-intensity quantitative phase microscopy with annular illumination. Sci. Rep. **7**(1), 7622–7654 (2017)
36. P. Wang, Y. Liu, SEMA: Secure and efficient message authentication protocol for VANETs. IEEE Syst. J. **15**(1), 846–855 (2021)
37. Y. Zhou, L. Tian, C. Zhu, X. Jin, Y. Sun, Video coding optimization for virtual reality 360-degree source. IEEE J. Sel. Top. Signal Process. **14**(1), 118–129 (2020)
38. M. Yang, A. Sowmya, An underwater color image quality evaluation metric. IEEE Trans. Image Process. **24**(12), 6062–6071 (2015)
39. Y. Tang, Z. Wang, J. Fang, Parameters identification of unknown delayed genetic regulatory network by a switching particle swarm optimization algorithm. Expert Syst. Appl. **38**, 2523–2535 (2011)
40. Y. Yang, J. Yao, C. Wang, Y. Gao, Q. Zhang, S. An, W. Song, New pore space characterization method of shale matrix formation by considering organic and inorganic pores. J. Nat. Gas Sci. Eng. **27**(11), 496–503 (2015)

Publisher's Note