Vehicular task scheduling strategy with resource matching computing in cloud-edge collaboration

Fangyi Hu | Lingling Lv | TongLiang Zhang | Yanjun Shi

Department of Mechanical Engineering, Dalian University of Technology, Dalian, China

Correspondence
Yanjun Shi, Mechanical Engineering Department, Dalian University of Technology, Dalian, China. Email: syj@ieee.org

Funding information
China National Key Research and Development Program, Grant/Award Number: 2018YFE0197700

Abstract
In future transportation, on board unit (OBU) is a key component of connected vehicles with limited computing resources, and may not tackle the heavy computing burden from V2X networks. For these cases, we herein employ multi-access edge cloud (MEC) and remote cloud to schedule the OBUs’ tasks. This schedule tries to minimise the total completion time of all tasks and the number of computing units of the MEC server. We first introduce a multi-objective optimisation model considering the tasks and cloud-edge collaboration. Then, we propose a task scheduling strategy considering the resource matching degree for this model. In this strategy, we propose an improved hybrid genetic algorithm and employ the resource matching measure between the tasks and computing units in terms of computing, storage and network bandwidth resources to obtain better solutions for generations. The numerical results showed the effectiveness of our strategy.

1 | INTRODUCTION

In recent years, the automotive industry is facing critical and tremendous changes. On board units (OBUs) provide us with a powerful platform for many innovative mobile applications, such as speech recognition, natural language processing, computer vision, machine learning, augmented reality, online planning and decision-making, etc. [1] They usually require abundant computing resources [2]. However, due to limitations in weight, size, and battery life, OBUs are short of computing resources [3]. Remote cloud with powerful computing and storage capabilities can replace computation-poor OBUs and provide abundant computing resources for these computation-intensive services [4]. By scheduling computation-intensive tasks on OBUs to the remote cloud servers with powerful computing and storage capabilities, the task scheduling strategy can improve the OBU characterised by weak computing capabilities and limited battery life. However, due to the long-distance transmission and the congestion of the core network [5], the remote cloud cannot meet the low latency requirements of the Internet of Vehicles (IoV).

As a key component of 5G communication technology, the multi-access edge cloud (MEC) server is usually deployed near intelligent base stations to provide low-latency services for vehicles within the coverage of the base station because of the avoidance of core networks [6]. As a constrained optimisation problem, the commonly used solution methods for task scheduling problems include chemical reaction algorithm [7], particle swarm algorithm [8], genetic algorithm [9], etc. There are a few studies on the task scheduling problem of MEC. In [10], the authors model the problem of computation offloading and resource allocation as a single-objective constrained optimisation to maximise the profit of the system including the terminal layer, edge computing layer and central cloud layer, and proposed a migratory bird algorithm based on simulated annealing. In [11], the authors propose a simulated annealing-based bee algorithm to minimise the energy cost of a distributed green cloud provider by specifying the running speed of each server and the number of powered-on servers in each green cloud. In [12], the authors propose a task scheduling mechanism that takes into account the computing and transmission energy consumption of the task to minimise the task scheduling system energy consumption in 5G network and obtain the task scheduling scheme with the minimum energy consumption under the constraints of wireless bandwidth and delay. In [13], from the perspective of users, the authors take the cost of task transmission and computation as the total cost, model the MEC computational task scheduling problem as an NP-hard mixed integer programming problem, and propose an iterative algorithm to minimise the time delay and total cost. In [14], considering radio resource allocation and computing resource allocation, the authors model the problem of minimising energy...
consumption of smart mobile devices as a convex model and propose a heuristic algorithm based on the Gini coefficient to minimise the energy consumption of smart mobile devices on the premise of meeting the task delay constraint.

Although the existing task scheduling strategies have good performance in reducing the total completion time of all tasks and ensuring reliability, they did not consider the full utilisation of the cloud-edge collaboration, which leads to the inability to further reduce the total completion time of all tasks. Therefore, in this study, we jointly consider the OBU, MEC and remote cloud, and propose a cloud-edge collaborative task scheduling strategy. The main contributions of this study can be summarised as follows:

1. A cloud-edge collaborative task scheduling model is established under the time delay constraint of each task and the constraints of storage resources and network bandwidth resources of computing units.
2. Task classification method, computing unit evaluation method and resource matching computing method of task scheduling scheme are proposed to improve the resource matching degree of task and computing unit, and further reduce the total completion time of all tasks.
3. An improved hybrid genetic algorithm (HGA) is proposed to solve the task scheduling model of cloud-edge collaboration and obtain the task scheduling scheme.

The rest of this study is structured as follows: we describe the system model and formulate the studied problem in Section 2; in Section 3, we propose an improved HGA approach to reduce the total completion time of all tasks and the number of required computing units, maximise the matching degree of resources between tasks and computing units, followed by performance evaluation through the numerical experiment in Section 4; finally, we conclude this study in Section 5.

2 | THE VEHICULAR TASK SCHEDULING PROBLEM

2.1 | The vehicular task scheduling problem description

In this study, we consider a 5G multi-user cellular network with edge cloud and remote cloud as shown in Figure 1. In this system, a single remote cloud, a single MEC server and some connected vehicles are in the communication range of the base station. Each vehicle has a computing task to compute T = {Ti | i = 1, 2, ..., N} and each task has five attributes Ti = {c1, s1, b1, l1, f1}, where c1 represents the CPU cycles required to compute task i, s1 represents the required data size, c1 is proportional to s1, and c1 can be measured offline [16], b1 represents the required bandwidth, l1 represents the time delay constraint, and f1 is the classification.

Due to the limitation of computing resources and battery life of the OBU, cloud-edge collaborative task scheduling is a significant approach to improve battery efficiency and computing performance for the OBU. However, considering the limited computing resources of the MEC server, it is difficult to meet the performance requirements only by scheduling the computing tasks to the MEC server. Therefore, for the computation-intensive task, we consider three methods. The first method is to execute locally on the OBU with limited computing resources. The second is to schedule the computing task to the MEC server and then to the remote cloud server through the wired network.

Besides, it is necessary to further explain the task scheduling process of cloud-edge collaboration. In this article, as shown in Figure 1. Firstly, the local software defined network (SDN) server collects (1) resource requirements and time delay constraints of each task; (2) resources owned by each computing unit. Secondly, the local SDN server solves the task scheduling problem through the task scheduling strategy. Finally, the local SDN server notifies the OBUs to send the task to the corresponding MEC server or remote cloud server for computing. For clarity, the main notations in this study are summarised in Table 1.

2.2 | The cloud-edge collaborative model building

2.2.1 | Computing task

There are N vehicles within the communication coverage of the base station. Each vehicle has a computing task to compute T = {Ti | i = 1, 2, ..., N} and each task has five attributes Ti = {c1, s1, b1, l1, f1}, where c1 represents the CPU cycles required to compute task i, s1 represents the required data size, c1 is proportional to s1, and c1 can be measured offline [16], b1 represents the required bandwidth, l1 represents the time delay constraint, and f1 is the classification.

2.2.2 | Computing unit

There are M computing units U = {Uj | j = 1, 2, ..., M} available for the tasks to schedule, in which U1 represents the OBU, U2 represents the remote cloud server and U3 – UM represent M - 2 virtual machines of the MEC server. Each computing unit can be represented as Uj = {Cj, Sj, Bj, Fj}, where Cj represents the CPU frequency of computing unit j, Sj represents the memory size, Bj represents the bandwidth and Fj is the classification.

2.2.3 | Simplification of completion time

The transmission time of a task is the ratio of the data size to the transmission rate, which depends on the bandwidth
allocated by the base station for the vehicle. To simplify the problem model, we transform the optimisation of task transmission time into the satisfaction of the bandwidth constraint. The specific approach is: each task needs to complete transmission within the specified time, tasks with different data sizes need different bandwidth allocation to achieve the time, $b_i$ depends on $c_i$, and the total bandwidth $B_j$ of each computing unit is limited. Therefore, the task scheduling scheme needs to meet the bandwidth constraint, and the objective function can only consider the task computing time.

Each computing unit computes each task sequentially and computing time for each task is:

**FIGURE 1** The architecture of task scheduling in cloud-edge collaboration

| Notations | Definition |
|-----------|------------|
| $N$       | Number of tasks/vehicles |
| $M$       | Number of computing unit |
| $c_i, B_i, f_i$ | CPU cycles, data size, bandwidth, time constraint and classification of task $T_i$ |
| $C_j, S_j, B_j, F_j$ | CPU frequency, memory size, bandwidth and classification of computing unit $U_j$ |
| $t_{ij}^1$ | Computing time for task $i$ scheduled to computing unit $j$ |
| $t_{ij}^2$ | Completion time for task $i$ scheduled to computing unit $j$ |
| $T_j$ | The total completion time of the tasks scheduled to computing unit $j$ |
| $O_{ij}$ | The number of tasks scheduled to the computing unit $j$ that need to be computed before task $i$ |
| $O_j$ | The number of tasks scheduled to the computing unit $j$ |
| $S$ | The size of the population |
| P1, P2 | The chromosome |
The completion time of the tasks is:

\[ t_{ij}^r = \frac{c_i}{C_j} \]  

1. Resource matching computing. The clustering algorithm is used to solve the matching degree between tasks and computing units;
2. Fitness function representation. The total completion time of all tasks is used as the evaluation index of the task scheduling scheme;
3. Population initialisation;
4. Crossover. The idea of extreme greed in adaptive large neighbourhood search algorithms is used for the crossover operation;
5. Selection. The roulette selection algorithm is used to select the remaining chromosomes;
6. Mutation.

2.2.4 Constrained optimisation problem

The cloud-edge collaborative scheduling problem of IoV is a multi-objective, multi-constraint and complex comprehensive scheduling process [17]. This study addresses the scheduling problem of tasks that need to be computed by the OBU in cloud-edge collaboration. The objective of this work is to minimise the total completion time of all tasks and the number of computing units, maximise the matching of resources between tasks and computing units. The constraint conditions are the time delay constraint of each task and the limited memory sizes and bandwidth of computing units.

The multi-constraint and multi-objective optimisation model of the task scheduling problem in this study can be expressed by the following equations:

\[
F_1(X) = \min \left( \max \left( T_j \right) \right) \quad \forall j = 1, 2, ..., M \\
 t_{ij}^r < l_i \\
\sum_{i=1}^{O_{j+1}} b_i < B_j \\
\sum_{i=1}^{O_{j+1}} s_i < S_j
\]  

1. Resource matching computing. The clustering algorithm is used to solve the matching degree between tasks and computing units;
2. Fitness function representation. The total completion time of all tasks is used as the evaluation index of the task scheduling scheme;
3. Population initialisation;
4. Crossover. The idea of extreme greed in adaptive large neighbourhood search algorithms is used for the crossover operation;
5. Selection. The roulette selection algorithm is used to select the remaining chromosomes;
6. Mutation.

3 THE IMPROVED HGA FOR VEHICULAR TASK SCHEDULING

For the cloud-edge collaborative scheduling problem in IoV, a HGA is designed to solve it. After solving the HGA, the OBU's decide whether to schedule the tasks to edge cloud or remote cloud or leave the tasks locally based on the task scheduling scheme. The HGA proposed in this study is divided into six steps:

1. Determine the number of cluster centres K;
2. Randomly select K tasks as the initial clustering centre;
3. Equation (10) was used to calculate the distance between each task and each cluster centre, and each task was assigned to the nearest cluster centre;
4. Judge whether the cluster centre needs to be changed and re-determine the cluster centre; and
5. Determine whether the clustering centre has been changed. If the change is very large, repeat steps 3 and 4 until no tasks are reassigned, no changes occur in the clustering centre and the local error sum of squares is minimised.

3.1.2 Computing unit evaluation

According to the main resource types of the computing unit, the computing units are divided into calculation type, storage type and network bandwidth type.

Step 1: Normalise the resources owned by the computing unit

\[ NC_j = \frac{M \times C_j}{\sum_{i=1}^{M} C_j} \]  
\[ NS_j = \frac{M \times S_j}{\sum_{i=1}^{M} S_j} \]  
\[ NB_j = \frac{M \times B_j}{\sum_{i=1}^{M} B_j} \]  

Step 2: For each computing unit \( j \), the type of the computing unit is determined according to the resource type corresponding to the larger value of \( NC_j \), \( NS_j \) and \( NB_j \), and then the vector \( F_j \) corresponding to each computing unit \( j \) is determined.

3.1.3 Resource matching computing

Maximise the matching of resources between the tasks and the computing units:

\[ F_2(X) = \max \left( \sum_{j=1}^{N} \left( \sum_{k=1}^{Q_j} F_{jk}^T \right) \right) \]  

3.2 Fitness function representation of the HGA

Fitness function is a method used in the genetic algorithm to evaluate the properties of the solution, and its value is a key index to determine whether the current solution is inherited to the next generation. In genetic algorithms, a small fitness value corresponds to a poor solution, and a large fitness value corresponds to a high-quality solution. Therefore, in the iterative process, the solution with a small fitness value will be gradually eliminated, and the solution with a large fitness value will be retained, resulting in the optimal solution of the problem. The fitness function is expressed as:

\[ F = \frac{F_2(X)}{F_1(X)} \]  

3.3 The initialisation of the HGA

The tasks of the OBU are scheduled to multiple computing units, so the solution to the problem consists of the order in which the tasks of the multiple computing units are computed. Here, a task list divided into M blocks is used to represent the tasks scheduled to each computing unit and their computing order. As shown in Figure 1.2, there are three computing units in total. The task computing order of the first computing unit is 2-4-5, the task computing order of the second computing unit is 3-7-1, and the task computing order of the third computing unit is 6-8.

First, generate a random arrangement of natural numbers one to \( N \) (representing tasks), and then take the numbers from the zeroth position in a sequence. According to the resource constraints of the computing unit and the time constraints of the task, the tasks are scheduled to \( M \) computing units in sequence. When the current computing unit cannot meet the constraints, the task is assigned to the next computing unit. After the above operations, a chromosome is finally encoded that is, the initial solution of the task scheduling problem is achieved. Repeat the above operation steps \( P - 1 \) times to generate an initial population with an individual size of \( P \).

3.4 The crossover of the HGA

The crossover operator is the main method to generate new chromosomes in a genetic algorithm. The basic principle is to exchange a portion of the gene sequence between two selected chromosomes to form two new chromosomes. The commonly used methods are single-point crossover, multi-point crossover, uniform crossover, arithmetic crossover, and compound crossover. Based on the thoughts of the greedy algorithm and recombination crossover algorithm, an optimal crossover operator is designed in this study. The operator mainly includes four processes: chromosome selection, computing unit selection, gene deletion and gene addition. The specific operation process of cross operation is as follows:

3.4.1 Chromosome selection

In a population with an individual size of \( P \), the two chromosomes are selected in turn for crossover operation without replacement, forming \( p \) new chromosomes, and the population size becomes \( 2P \).

3.4.2 Computing unit selection:

From the two chromosomes P1 and P2 are shown in Figure 2.3, the first computing unit with tasks 2, 4 and 5, and the second computing unit with tasks 1 and 5 are randomly selected.
3.4.3  |  Gene deletion

As shown in Figure 3.4, tasks 1 and 5 are deleted from chromosome P1, and tasks 2, 4 and 5 are deleted from chromosome P2.

3.4.4  |  Gene addition:

The process of gene addition is the process of gene recombination. In this process, chromosome P1 needs to add tasks 1 and 5, and chromosome P2 needs to add tasks 2, 4 and 5, thus forming two new chromosomes. The specific method is as follows: for chromosome P1, first, select task 1, and then judge its insertion point from left to right. The criterion is that the point with the maximum fitness value is the best insertion point under the premise of satisfying the constraints. After determining the insertion point of task 1, select the insertion point of task 5. Similarly, for chromosome P2, the insertion points are selected for tasks 2, 4 and 5 respectively. After the above operations, two new chromosomes are formed, that is, two new task scheduling schemes.

Algorithm 1 gives the pseudocode, including both the removal and insertion stage in this process.

Algorithm 1 The crossover operator

**Input:** P1: parent1; P2: parent2.

**Output:** O1: offspring one; O2: offspring two.

1: Random chose a computing unit U1, U2 from P1 and P2, respectively;
2: Remove the tasks in U2 from P1 and generate O1;
3: Remove the tasks in U1 from P2 and generate O2;
4: For a task t in U2 (U1);
5: For a computing unit in O1 (O2);
6: If the current computing unit adding task t satisfies the constraints then;
7: Insert t into each interval in sequence in the current computing unit and record the fitness values;
8: Else;
9: move to the next computing unit;
10: End if the last computing unit has been reached;
11: If there are no computing satisfying constraints then;
12: Insert task t into a newly added computing unit;
13: Else.
14: Insert task t into the position with a minimum fitness value;
15: End if all the tasks have been inserted into O1 (O2);
16: Return O1 (O2).

3.5  |  The selection of the HGA

The selection operation process of the genetic algorithm is the selection process of each chromosome in the population, selecting the chromosome with a higher fitness value. This process stimulates the law of survival of the fittest [18]. This study uses the roulette algorithm. Specific steps are as follows:

Step 1: Calculate the proportion of fitness, that is, the selection probability of each chromosome.

Step 2: Calculate the cumulative probability of each chromosome, which is equivalent to the span on the turntable. The larger the span, the easier it is to be selected.

Step 3: Randomly generate a decimal between 0 and 1, and select the chromosome corresponding to the smallest cumulative probability greater than the decimal.

Step 4: Repeat steps 2 and 3 until P chromosomes are selected, and the remaining P chromosomes are eliminated.

3.6  |  The mutation of the HGA

In nature, new species are created by a genetic mutation. Based on this idea, the genetic algorithm further increases the diversity of individuals in the population through the mutation operation of chromosomes. The basic idea is to change the order of some genes in chromosomes. In the process of generating new chromosomes, the crossover operator belongs to the global search process, while the mutation operator belongs to the local search process. Through the global search and the local search, the genetic algorithm has good search performance. The mutation operation is shown in Figure 4.5: computing units 1 and 3 are randomly selected, and tasks 2 and 7 are randomly selected from computing units 1 and 3, respectively. After the process of the chromosome mutation operation, if the task scheduling scheme corresponding to the new chromosome meets the constraint conditions, the old chromosome will be replaced with the new chromosome. Otherwise, the original chromosome will be retained.

![Figure 2](image-url)  The representation of the task scheduling scheme

![Figure 3](image-url)  The operation of computing unit selection
3.7 Task scheduling problem-solving steps

1. According to the number of resources required for each task, the computing tasks in the task set of the OBU are classified. Evaluate computing units based on the number of resources each computing unit has.

2. Determine the fitness function, that is, the matching degree of all computing tasks and the cell to which they are scheduled divided by the total delay time of all computing tasks.

3. Initialise P chromosomes, and the gene sequence of each chromosome represents the solution of a task scheduling problem, that is, the task assigned by each cell and the task calculation sequence.

4. Without putting them back, two chromosomes are sequentially selected for crossover operation, and the population size becomes 2P. First, randomly select the computing unit for each chromosome, secondly delete the tasks corresponding to the selected computing unit in the other chromosome, and finally insert the deleted tasks in sequence based on the greedy algorithm.

5. Select P chromosomes with higher fitness from the 2P chromosomes according to the roulette algorithm.

6. For each chromosome, the mutation is generated with a probability of \( \alpha \). For the chromosome with mutation, randomly select two computing units on the chromosome, and then randomly select a computing task exchange position in the two computing units to retain the chromosome with the highest fitness.

7. Repeat steps 4–6 until no significant fitness change occurs.

The HGA executes in a straight and simple way. Algorithm 2 presents the pseudocode of it. Also, the flow chart of HGA is shown in Figure 6. The task scheduling problem is a constraint optimisation problem. The average time complexity of the algorithm in this study is the square of the problem size (task/vehicle number).

Algorithm 2 The HGA

Input: \( P_c \): the probability of crossover; \( P_m \): the probability of mutation; \( S \): the size of the population; MI: the maximum iteration.
Output: the best chromosome.

1: Parameter setting;
2: Generate \( S \) initial feasible solutions using the chromosome initialisation strategy;
3: Calculate the fitness values of chromosomes;
4: Repeat;
5: Select two parents from the chromosomes randomly;
6: Generate a random number \( R \) between 0 and 1;
7: If \( R < P_c \) then;
8: Produce two offspring based on the crossover operator;
9: Else;
10: Take the two parents as two offspring;
11: Repeat \( S \) times and generate 2S offspring;
12: Repeat;
13: Select one offspring from the chromosome in sequence;
14: Generate a random number \( R \) between 0 and 1;
15: If \( R < P_m \) then;
16: Mutate the offspring based on mutation operator;
17: Else;
18: Do not execute mutation;
19: Repeat 2S times and generate 2S mutated offspring;
20: Select two parents randomly from 2S chromosomes using roulette algorithm and repeat \( S \) times;
21: End if the iteration generation MI is achieved;
22: Return the best chromosome B.

4 Computational Experiments

The configuration of the numerical experiment used in this study is as follows: 64-bit Windows 10 Professional operating system, 2.10 GHz Intel i5-9300 HF processor, 8G memory, 500G hard disk, and a Python language integrated development tool called PyCharm.

This study tests the performance of the improved HGA by numerical analysis when the number of tasks (vehicles) is 100, 200 and 400. The parameters of the improved HGA and the virtual machine configuration parameters of the OBU, the
FIGURE 6  Steps of the HGA. HGA, hybrid genetic algorithm

| Parameters | Definition                                              | Values                      |
|------------|---------------------------------------------------------|-----------------------------|
| $S$        | The size of the population                             | 100, 200 and 400            |
| $P_c$      | The probability of crossover                           | 0.9                         |
| $P_m$      | The probability of mutation                            | 0.1                         |
| MI         | The maximum iteration                                  | 200                         |
| $N$        | The number of tasks                                     | 100, 200 and 400            |
| $M$        | The number of computing unit                           | 6, 10 and 16                |
| $C_j$      | Computing resources of the computing units             | 10–300                      |
| $S_j$      | Storage resources of the computing units               | 10–500                      |
| $B_j$      | Network bandwidth resources of the computing units     | 20–600                      |
MEC server and the remote cloud server are shown in Table 2. Among them, set the $K$ value in the resource matching computing to three based on the number of task attributes; According to [19], the population size is selected as the number of tasks, the crossover and mutation probability are selected as 0.9 and 0.1.

When the number of tasks (vehicles) is 100, a total of five computing units are needed, namely, a remote cloud server and four virtual machines of the MEC server. When the number of tasks is 200, a total of nine computing units are needed, namely, a remote cloud server and eight virtual machines of the MEC server. When the number of tasks is 400, a total of 15 computing units are needed, namely, an OBU, a remote cloud server and 14 virtual machines of the MEC server. In the

![K-means clustering results: (a) 100 tasks, (b) 200 tasks and (c) 400 tasks](image)

**Figure 7** $K$-means clustering results: (a) 100 tasks, (b) 200 tasks and (c) 400 tasks

![Iteration diagram of the fitness value: (a) 100 tasks, (b) 200 tasks and (c) 400 tasks](image)

**Figure 8** Iteration diagram of the fitness value: (a) 100 tasks, (b) 200 tasks and (c) 400 tasks
operation process of the HGA proposed in this study, the optimisation results of the fitness value, the resource matching degree and total completion time of the task scheduling scheme corresponding to the chromosome with the highest fitness in each generation are shown in Figures 6–8, and the optimisation results of the task scheduling scheme corresponding to the chromosome with the highest fitness are shown in Table 3.

The objective function of the task scheduling problem in IoV is to minimise the total completion time of all tasks and the number of computing units required, maximise the matching of resources between the tasks and the computing units. Through the above simulation experiment, it can be seen that the resource matching degree has a positive correlation with the impact of task completion time. Besides, when the task volume is 100, the algorithm has a good optimisation effect; when the task volume reaches 200 and 400, the
algorithm also has a good optimisation effect for fitness value and resource matching degree. For the total completion time of all tasks, the optimisation effect is not obvious, but the final result is still within a reasonable range. Therefore, the numerical results show that the task scheduling scheme obtained by the HGA proposed in this study is effective for the vehicular task scheduling problem.

5 | CONCLUSION

This study addresses the task scheduling problem in IoV considering the cloud-edge collaborative characteristic. Under the constraints of time delay, computation, and network bandwidth resources, the computing method of each vehicular task is decided to minimise the total completion time of all tasks. This study provides an idea for computation offloading of IoV and promotes the deployment of real-time applications of IoV.

However, in the modelling process, we transform the optimisation of task transmission time into the satisfaction of bandwidth constraint, which simplified the processing of task transmission. In the algorithm part, although the iterative algorithm adopted was running on the SDN server, there was still a certain amount of time loss. In future research, we will consider the real communication process between the vehicle and the server, and use deep reinforcement learning to solve the task scheduling problem online.

REFERENCES

1. Khaliti, B., Hamdaoui, B., Guizani, M.: Extracting and exploiting inherent sparsity for efficient IOT support in 5G: challenges and potential solutions. IEEE Wirel. Commun. 24(5), 68–73 (2017)
2. Pan, Z., Zhang, Y., Sam, K.: Efficient motion and disparity estimation optimization for low complexity multiview video coding. IEEE Trans. Broadcast. 61(2), 166–176 (2015)
3. Wang, X., Ning, Z., Wang, L.: Offloading in internet of vehicles: a fog-enabled real-time traffic management system. IEEE Trans. Ind. Inform. 14(10), 4568–4578 (2018)
4. Wang, Z., Hao, W., Yan, L.: Cooperative scheduling of multi-remote and cloud resources: fine-grained offloading strategy for multithreaded applications. IET Commun. 14(10), 1632–1641 (2020)
5. Mao, Y., et al.: A survey on mobile edge computing: the communication perspective. IEEE Commun. Surv. Tutor. 19(4), 2322–2358 (2017)
6. Huang, X., et al.: Exploring mobile edge computing for 5G-enabled software-defined vehicular networks. IEEE Wirel. Commun. 24(6), 55–63 (2017)
7. Jun, L., et al.: Solving multi-area environmental/economic dispatch with Pareto-based chemical reaction optimization algorithm. IEEE/CAM J. Autom. Sin. 6(5), 1240–1250 (2019)
8. Zhi, L., et al.: Surrogate-assisted particle swarm optimization algorithm with Pareto active learning for expensive multi-objective optimization. IEEE/CAM J. Autom. Sin. 6(3), 838–849 (2019)
9. Yan, H., et al.: Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm. IEEE Trans. Syst. Man Cybern. Syst. 47(3), 517–530 (2017)
10. Hai, Y., Meng, Z.: Profit-maximised collaborative computation offloading and resource allocation in distributed cloud and edge computing systems. IEEE Trans. Autom. Sci. Eng., 1–11 (2020)
11. Hai, Y., et al.: Fine-grained resource provisioning and task scheduling for heterogeneous applications in distributed green clouds. IEEE/CAM J. Autom. Sin. 7(5), 1380–1393 (2020)
12. Zhang, K., et al.: Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks. IEEE Access. 4, 5896–5907 (2016)
13. Wang, K., et al.: Joint offloading and charge cost minimisation in mobile edge computing. IEEE Open J. Commun. Soc. 1, 205–216 (2020)
14. Zhao, P., et al.: Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing. IEEE Access. 5, 11255–11268 (2017)
15. Yang, L., et al.: Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans. Veh. Technol. 67(7), 6398–6409 (2018)
16. Chen, X., et al.: Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE Trans. Netw. 24(5), 2795–2808 (2016)
17. Taleb, T., et al.: On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration. IEEE Commun. Surv. Tutor. 19(3), 1657–1681 (2017)
18. Akhtar, M., Raza, M., Shafiq, M.: Role of batch size in scheduling optimization of flexible manufacturing system using genetic algorithm. J. Ind. Eng. Int. 15(1), 135–146 (2019)
19. Na, L., et al.: An effective order-aware hybrid genetic algorithm for capacitated vehicle routing problems in internet of things. IEEE Access. 7, 86102–86114 (2019)

How to cite this article: Hu F, Lv L, Zhang TL, Shi Y. Vehicular task scheduling strategy with resource matching computing in cloud-edge collaboration. IET Collab. Intel. Manuf. 2021;1–11. https://doi.org/10.1049/cim2.12023