An Adaptive Task Offloading Strategy in Vehicle-Edge Collaborative Environment

Chao Zhang, *Member, IEEE,* Hui Zhao, *Member, IEEE,* Yuhang Dong , *Member, IEEE*

***Abstract*—Due to the limited computational resources and coverage of edge servers, it will delay the task completion time and affect the user experience when there is a surge of vehicle task offloading requests or the vehicle is not in the coverage of the edge server. However, some vehicles on the road may have idle resources after meeting their own needs, and load imbalances among edge servers can lead to resource waste. To address this issue, this paper proposes an adaptive task offloading strategy in a vehicle-edge collaborative environment to enhance task offloading efficiency and reduce task completion time. First, based on the vehicle positions and the communication range of edge servers, the cooperative relationships among vehicles and between edge servers are considered to construct a task offloading model in the vehicle-edge collaborative environment, with the goal of minimizing task completion time. Second, an adaptive task offloading strategy is designed according to vehicle positions. When vehicles are outside the coverage of edge servers, an optimal service vehicle selection algorithm is proposed to offload tasks to the best service vehicle through vehicle collaboration. When vehicles are within the coverage of edge servers, a hybrid differential teaching optimization-based task scheduling algorithm is proposed to offload tasks to the optimal edge server through edge server collaboration. Finally, simulation experiments are designed and compared with existing task offloading methods in vehicular networks to verify the effectiveness of the proposed strategy.**

***Index Terms*—Edge computing, Internet of vehicles, Task offloading, Teaching optimization**

I. Introduction

With the rapid development of technologies such as wireless communication and artificial intelligence, the level of intelligence in vehicles is increasing, promoting the advancement of Internet of Vehicles (IoV) systems. In IoV systems, the computing demands for intelligent services such as autonomous driving, real-time navigation, and infotainment are rapidly escalating, with stricter requirements for low latency [1]. In the future, sixth-generation wireless communication technology will provide strong support for IoV's V2X (Vehicle-to-Everything) communication, which not only enables more advanced applications in IoV but also presents significant challenges for resource-limited intelligent vehicles [2].

In recent years, many studies have focused on task offloading techniques in IoV systems, aiming to offload tasks that are difficult or costly for vehicles to process onto cloud centers, thereby improving task execution efficiency [2]–[5]. However, for latency-sensitive tasks in IoV systems, traditional cloud-based task offloading can no longer meet the real-time requirements of intelligent vehicle applications. In contrast, deploying low-cost edge servers near users is an effective computation offloading solution.

In the new generation of IoV technology, deploying edge servers at base stations or roadside units can significantly reduce round-trip latency in data transmission, ensuring real-time responses to user requests [6]–[13]. However, edge server resources are often limited. When task offloading requests from vehicles surge or there is a high density of computation-intensive tasks in an area, edge server overload can occur, affecting service quality. Additionally, due to the limited coverage range of statically deployed edge servers, tasks cannot be offloaded to an edge server once a vehicle moves out of its coverage area. Meanwhile, some vehicles on the road may have idle resources available after meeting their own needs. Therefore, fully utilizing the idle resources of vehicles on the road to achieve vehicle-edge collaborative task offloading is of great research significance and value for improving task execution efficiency in IoV environments.

Based on the above analysis, to make full use of the resources of vehicles and edge servers in the system and to reduce task completion time, this paper proposes an adaptive task offloading strategy in a vehicle-edge collaborative environment. This strategy adaptively selects the task offloading algorithm based on vehicle location and the communication range of edge servers to achieve optimal task offloading. When vehicles are outside the coverage range of an edge server, a best-service vehicle selection algorithm is proposed to utilize vehicle-to-vehicle cooperation for offloading tasks to the optimal service vehicle. When vehicles are within the edge server's coverage range, a task scheduling algorithm based on hybrid differential teaching optimization is proposed, which considers cooperation between edge servers to offload tasks to the most suitable edge server, achieving balanced load distribution among edge servers. Finally, experimental results demonstrate that the proposed strategy can effectively reduce task completion time. The main contributions of this paper are as follows:

(1)To improve task execution efficiency and fully utilize the resources within the IoV system, we consider the collaborative relationship between vehicles and edge servers. With the objective of minimizing task completion time, we have established a task offloading model in a vehicle-edge collaborative environment and designed an adaptive task offloading strategy based on vehicle location.

(2)When a vehicle is outside the coverage area of an edge server, a best-service vehicle selection algorithm is proposed,

This paper was produced by the IEEE Publication Technology Group. They are in Piscataway, NJ.

Manuscript received April 19, 2021; revised August 16, 2021.

which offloads tasks to the optimal service vehicle through cooperation between vehicles. To ensure task latency and stability of inter-vehicle communication, the algorithm considers only idle vehicle resources within a certain range and establishes a service vehicle task waiting queue to ensure orderly task execution.

(3)When a vehicle is within the coverage area of an edge server, a task scheduling algorithm based on hybrid differential teaching optimization is proposed. This algorithm considers cooperation between edge servers and, based on the KKT conditions, obtains the optimal location and proportion for task offloading to the edge server. This achieves load balancing among edge servers and reduces task completion time.

II. RELATED WORK

In recent years, significant research has been conducted on vehicle task offloading in vehicle edge computing (VEC) environments. For example, Liu et al. [6] proposed a dependency-aware task offloading scheme for collaborative computing among vehicles, edges, and clouds to obtain optimal offloading decisions. Zhang et al. [7] proposed a new task offloading method based on a simulated annealing mechanism, considering the current road density. Zhu et al. [8] proposed a distributed power allocation scheme to optimize energy consumption and latency in the VEC environment, taking into account the randomness of task arrivals, uncertainty of channel conditions, and vehicle mobility. Huang et al. [9] used greedy algorithms, simulated annealing, and heuristic rules to determine the optimal task offloading location and execution order on edge servers. Qian et al. [10] considered multiple indicators such as task acceptance rate, task completion rate, average completion time, and computing resource utilization and proposed a fine-grained algorithm to complete task scheduling within edge servers. Hossain et al. [11] proposed a dynamic task offloading algorithm based on non-cooperative game theory to offload tasks to edge or cloud servers, maximizing benefits while reducing task execution time and failure rates. Tang et al. [12] designed new computing and communication models to optimize the average response time of applications in VEC. Zhu et al. [13] proposed a task offloading decision scheme based on an improved non-dominated sorting genetic algorithm for the task offloading decision problem in intelligent transportation cloud-edge-end collaborative computing scenarios. All these studies leverage edge server cooperation to complete vehicle task offloading. In our previous research [14], we also explored task scheduling in edge-cooperative environments. Although edge servers can improve task offloading efficiency in IoV environments, as offloading requests increase, limited edge server resources may lead to load imbalances among servers, which can decrease task execution efficiency. Additionally, edge servers have limited coverage areas, so vehicles outside the coverage area cannot offload tasks to the edge server, affecting user experience. Therefore, considering both edge server and vehicle computing resources is of great significance for improving task offloading in IoV.

In VEC environments, in addition to edge server computing resources, some vehicles on the road may have idle resources after meeting their own needs. Using these idle resources to assist edge servers can help improve task offloading efficiency. Due to the mobility of vehicles, the time that vehicles stay within the coverage area of roadside units (RSUs) is limited. When vehicles move out of the RSU communication range, tasks can no longer be directly offloaded to edge servers, impacting service quality. Ming et al. [15] proposed a dynamic, adaptive distributed and centralized deep reinforcement learning method to use both vehicles and edge servers as computing service nodes to respond to different task demands and reduce system latency. Chen et al. [16] used nearby RSUs and vehicles to provide edge computing services, employing game theory concepts to incentivize nearby vehicles to share computing resources, ensuring vehicle safety and minimizing task latency. Gong et al. [17] proposed a stepwise computing offloading algorithm that decomposes applications into dependent subtasks to minimize the time interval between task offloading and task execution. Li et al. [18] proposed a new scalable modulation-based task offloading strategy, selecting the optimal moving vehicle as a relay node to assist in offloading tasks to the edge server based on the current network topology, thereby minimizing energy consumption while ensuring latency and transmission quality. To reduce latency in intelligent vehicular networks, Pang et al. [19] proposed a relay vehicle selection and reputation management algorithm to choose reliable relay vehicles while managing vehicle trustworthiness. Zhang et al. [20] addressed potential trust issues among vehicles in V2V and V2I communication, proposing a reputation management system based on an improved three-valued subjective logic algorithm for inter-vehicle reputation management, using blockchain technology to ensure information security in vehicular networks. The above studies use vehicle computing or communication resources to assist in task offloading but do not fully consider factors such as vehicle location, speed, and direction. Additionally, multi-hop transmission among vehicles in dynamic mobile environments poses certain risks, and further research is needed on how to ensure communication stability and improve task offloading efficiency.

Unlike the above studies, this paper further improves task execution efficiency by considering both edge collaboration and inter-vehicle collaboration. It proposes an adaptive task offloading algorithm in a vehicle-edge collaborative environment, making optimal task offloading decisions based on the resources of both vehicles and edge servers.

III. VEHICLE-EDGE COLLABORATIVE TASK OFFLOADING MODEL

This paper considers a scenario where vehicles are traveling on a highway, as shown in Fig. 1. It is assumed that K RSUs (Roadside Units) are evenly distributed along the roadside, with each RSU equipped with a mobile edge server (MEC) via a wired link. The set of RSUs is denoted as $M = \{M_1, M_2, \ldots, M_K\}$, and the distance between adjacent RSUs along the roadside is represented as . Vehicles travel freely on a two-way road and are categorized into request vehicles (RV) and service vehicles (SV). It is assumed that request vehicles

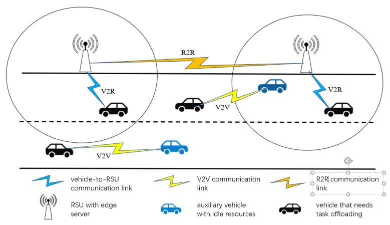

R2R
V2R
V2V
V2R
V2V
vehicle-to-RSU communication link
V2V communication link
R2R communication link
RSU with edge server
auxiliary vehicle with idle resources
vehicle that needs task offloading

Fig. 1. Vehicle-Edge Collaborative Task Offloading Architecture.

generate computation-intensive tasks, while service vehicles, in addition to completing their own tasks, have idle resources available, meaning they have spare resources for use after meeting their own needs. The set of request vehicles is denoted as $S = \{S_1, S_2, \ldots, S_N\}$, and the computational resources of vehicle S_i are denoted as f_i. Each vehicle generates tasks following a Poisson distribution with a parameter λ. A computation task is represented as $X_i = \{c_i, d_i\}$, where c_i denotes the number of CPU cycles required per bit of data, and d_i denotes the data size of the generated task. The variable $\delta_i\,(0 \leq \delta_i \leq 1)$ is defined as the proportion of the task offloaded to other nodes for execution. The $(1-\delta_i)\,d_i$ portion of task X_i is processed locally, while the remaining portion can be offloaded to an edge server or a service vehicle. However, the same task cannot be offloaded to both the edge server and a service vehicle simultaneously.

A. Communication Model

V2R (Vehicle to RSU) Communication Model: When a vehicle is within the coverage range of an RSU, the vehicle's information is automatically collected by the RSU, and tasks are offloaded to the RSU through a Vehicle-to-Infrastructure (V2R) communication link. Assuming W_i is the bandwidth resource allocated by the RSU to vehicle i, the transmission rate of the communication link between the vehicle and RSU can be expressed as:

$$R_i^{RSU} = W_i \log_2\left(1 + \frac{p_i |h|^2}{\mu_0 (d_{s_i,RSU})^{\vartheta}}\right) \tag{1}$$

where p_i denotes the transmission power of the vehicle, h is the channel gain between the vehicle and RSU, $d_{s_i,RSU}$ represents the distance between the vehicle and RSU, ϑ is the path loss exponent, and μ_0 represents Gaussian white noise power.

Assuming that at time t, the position of request vehicle S_i is (S_i^x, S_i^y), and the coverage radius of the edge server is d_{MEC}, the vehicle S_i is considered to be within the RSU's communication range and can establish a V2R communication link when $d_{s_i,RSU} \leq d_{MEC}$. $d_{s_i,RSU}$ can be represented as:

$$d_{s_i,RSU} = \sqrt{(S_i^x - M_k^x)^2 + (S_i^y - M_k^y)^2} \tag{2}$$

V2V (Vehicle to Vehicle) Communication Model: When a request vehicle needs to offload a task to a service vehicle, there is a connection delay prior to data transmission to complete information broadcasting and establish the connection, typically treated as a constant. Assuming the total bandwidth of the service vehicle is W, and that the request vehicles equally share the bandwidth during data transmission, the transmission rate between request vehicle S_i and service vehicle S_j can be expressed as:

$$R_i^j = \left(\frac{W}{m_j}\right) \log_2\left(1 + \frac{p_i |h|^2}{\mu_0 \left(d_{S_i,S_j}\right)^{\vartheta}}\right) \tag{3}$$

where m_j is the number of request vehicles choosing to offload tasks to service vehicle S_j, and d_{S_i,S_j} is the distance between request vehicle S_i and service vehicle S_j. The communication range of request vehicles is d_{VEH}, and a V2V communication link can be established between the vehicles when $d_{S_i,S_j} \leq d_{VEH}$. d_{S_i,S_j} is represented as:

$$d_{S_i,S_j} = \sqrt{\left(S_i^x - S_j^x\right)^2 + \left(S_i^y - S_j^y\right)^2} \tag{4}$$

The connection duration T_{S_i,S_j}^{con} between request vehicle S_i and service vehicle S_j can be expressed as:

$$T_{S_i,S_j}^{con} = \frac{d_{VEH} - a d_{S_i,S_j}}{\left|v_{S_i} - b v_{S_j}\right|} \tag{5}$$

where v_{S_i} and v_{S_j} represent the current speeds of request vehicle S_i and service vehicle S_j, respectively. Based on the position and speed of the vehicles, there are four possible cases: $a = 1, b = 1$ indicates that S_i and S_j are moving in the same direction, with the lead vehicle moving faster; $a = -1, b = 1$ indicates they are moving in the same direction, with the trailing vehicle moving faster; $a = 1, b = -1$ indicates they have passed each other and are now moving in opposite directions; $a = -1, b = -1$ indicates they are moving in opposite directions and have not yet passed each other.

B. Computation Model

Local computation time: Assume that the task offloading ratio for vehicle S_i is δ_i, and the local execution speed is f_i^{loc}. The local execution time of task X_i can be expressed as:

$$t_i^{loc} = \frac{(1-\delta_i)\, d_i c_i}{f_i^{loc}} \tag{6}$$

Service vehicle computation time: Considering that on-board computing resources are limited and cannot handle tasks in parallel, tasks offloaded to service vehicle S_j need to enter a waiting queue for execution. The task will only be processed once the previous task is completed. The execution time of task X_i offloaded to service vehicle S_j includes task transmission time, task waiting time, and task execution time, which can be expressed as:

$$t_i^{sv} = t_j^{wait} + t_i^{tran} + \frac{\delta_i d_i c_i}{f_j^{idl}} \tag{7}$$

where f_j^{idl} is the idle resource available for service vehicle S_j, t_j^{wait} represents the waiting time in the queue, and t_i^{tran} is the task transmission time.

$$t_i^{tran} = \frac{\delta_i d_i}{R_i^j} \tag{8}$$

During the transmission of task X_i, the local computation part can be executed first, so the local and offloaded execution are parallel. The total execution time is the maximum of both. Assuming that the execution result is small, the data return time is negligible. Therefore, the total execution time of task X_i offloaded to the service vehicle is:

$$t_i = \max\left\{t_i^{loc}, t_i^{sv}\right\} \tag{9}$$

Since the vehicle connection time is limited, it must also ensure that $t_i \leq T_{S_i,S_j}^{con}$, otherwise, if the communication connection is interrupted and the task execution fails, the task will be re-executed locally, leading to additional costs.

Edge server computation time: When the user is within the coverage range of the edge server, the task can be offloaded to the edge server for execution. Since there are wired links between edge servers, tasks can be forwarded to reduce the load on the edge server and improve task execution efficiency. Therefore, when tasks are offloaded to the edge server, the execution time includes task upload time, forwarding time between edge servers, and task execution time. The computing resources allocated by the edge server to the offloaded task are f_i^{RSU}, and the total execution time of the task offloaded to the edge server is:

$$t_i^{RSU} = t_i^{up} + t_i^{forw} + \frac{\delta_i d_i c_i}{f_i^{RSU}} \tag{10}$$

When $d_{s_i,RSU} \leq d_{MEC}$, it indicates that the task is within the RSU coverage range and can be offloaded to the edge server. The task upload time is:

$$t_i^{up} = \frac{\delta_i d_i}{R_i^{RSU}} \tag{11}$$

Task forwarding time occurs between edge servers via wired links. e_i represents the number of hops for task forwarding between edge servers, and the transmission rate is fixed at R_{RSU}. The forwarding time can be expressed as:

$$t_i^{forw} = \frac{e_i d_i \delta_i}{R_{RSU}} \tag{12}$$

Similar to offloading to service vehicles, tasks can also be executed in parallel at the local and edge servers. Therefore, the task execution time is:

$$t_i = \max\left\{t_i^{loc}, t_i^{RSU}\right\} \tag{13}$$

C. Problem Definition

As mentioned above, the processing delay of task X_i for request vehicle S_i mainly consists of the maximum value between local execution time and the execution time of the offloaded portion. Assume that the task can only be offloaded to one node. Let $o_{sv} = 1$ indicate that the task is offloaded to the service vehicle for execution, and $o_{RSU} = 1$ indicate that the task is offloaded to the edge server for execution, with the constraint $o_{sv} + o_{RSU} = 1$. The execution time of task X_i can be expressed as:

$$t_i = \max\left\{t_i^{loc}, o_{sv} t_i^{sv} + o_{RSU} t_i^{RSU}\right\} \tag{14}$$

Therefore, the task offloading optimization problem in the vehicle-edge collaborative environment can be defined as:

$$\min\left(\sum_{i=1}^{N} t_i\right) \tag{15a}$$
$$0 \leq \delta_i \leq 1, \quad \forall i \in N \tag{15b}$$
$$0 \leq f_i^{RSU} \leq F_{RSU}, \quad \forall i \in N \tag{15c}$$
$$0 \leq W_i \leq W_{RSU}, \quad \forall i \in N \tag{15d}$$
$$\sum_{i=0}^{N} f_i^{RSU} \leq F_{RSU}, \quad \forall i \in N \tag{15e}$$
$$\sum_{i=0}^{N} W_i \leq W_{RSU}, \quad \forall i \in N \tag{15f}$$
$$o_{sv} + o_{RSU} = 1 \tag{15g}$$

Constraint (15b) represents the task offloading ratio limitation. Constraints (15c) and (15e) ensure that the resources allocated by the edge server to offloaded tasks do not exceed its own computational resource limits. Constraints (15d) and (15f) ensure that the communication resources allocated to the task transmission channels by the edge server do not exceed the maximum bandwidth limit. Constraint (15g) ensures that at any given time, a task can only be offloaded to either the service vehicle or the edge server for execution.

IV. Adaptive Task Offloading Strategy

According to constraint (15g), problem (15a) can be transformed into:

$$t_i = \min\left\{\max\{t_i^{loc}, t_i^{sv}\}, \max\{t_i^{loc}, t_i^{RSU}\}\right\} \tag{16}$$

Problem (16) involves selecting the node with the minimum task execution delay for offloading. This can be solved by solving problems (9) and (13) separately. For the former, when the vehicle is outside the edge server's coverage range, this paper proposes an optimal service vehicle selection algorithm to ensure communication stability while minimizing task execution time. For the latter, this paper proposes a task scheduling algorithm based on hybrid differential evolution optimization, which determines the task offloading ratio and the offloading location. The resource allocation ratio for the offloaded task by the edge server is obtained using the Karush-Kuhn-Tucker (KKT) conditions. The two algorithms are iteratively solved alternately to ultimately obtain the optimal offloading decision.

A. Optimal Service Vehicle Selection Algorithm

Through analysis, it is known that when $t_i^{loc} = t_i^{sv}$, the task execution time is minimized. Based on this, the task offloading ratio can be obtained as follows:

$$\frac{(1-\delta_i) d_i c_i}{f_i^{loc}} = \frac{\delta_i d_i c_i}{f_j^{idl}} + t_j^{wait} + \frac{\delta_i d_i}{R_i^j} \tag{17}$$

where t_j^{wait} is the waiting time before the task starts executing on the current service vehicle.

The task offloading ratio δ_i is derived as:

$$\delta_i = \frac{\frac{d_i c_i}{f_i^{loc}} - t_j^{wait}}{\frac{d_i c_i}{f_i^{loc}} + \frac{d_i c_i}{f_j^{idl}} + \frac{d_i}{R_i^j}} \tag{18}$$

All vehicles can be categorized into request vehicles and service vehicles. The latter, in addition to handling local computation tasks, has some idle resources available. Therefore, request vehicles can offload tasks to service vehicles for execution. Due to the limited onboard computing resources, which are not conducive to resource allocation functions, tasks offloaded to service vehicles must first enter a waiting queue, awaiting the completion of earlier tasks, as shown in Fig. 2.

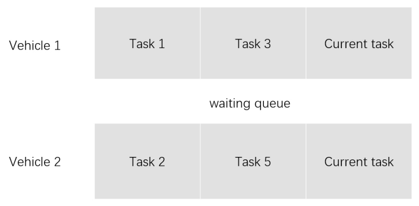


Fig. 2. Task execution waiting queue on service vehicle.

Additionally, vehicles are moving at high speeds on the road and may also be traveling in opposite directions. The communication range of vehicles is very limited, and the stability of the V2V communication connection is crucial. Therefore, this paper only considers service vehicles within the one-hop range of the requesting vehicle. The task offloading ratio is calculated based on equation (18) to obtain both the local execution time and the service vehicle's execution time. Then, the maximum connection time between the requesting vehicle and the service vehicle is estimated, and under the constraint of connection time, the service vehicle with the lowest execution delay is selected.

Best Service Vehicle Selection Algorithm (VSA) is shown in Algorithm 1.

Algorithm 1 Optimal Service Vehicle Selection Algorithm (VSA)

```
1:  Initialize: vehicle set $S$
2:  for each vehicle $S_i \in S$ do
3:      Search for the set of service vehicles $\Omega$ within its communication range
4:      for each service vehicle $S_j \in \Omega$ do
5:          Compute the communication retention time $T^{\text{con}}_{S_i,S_j}$ and task offloading ratio $\delta_i$ based on Equations (5) and (18)
6:          Initialize the task waiting time $t_j^{\text{wait}}$
7:          for each task $X_j$ in $S_j^{\text{wait}}$ do
8:              $t_j^{\text{wait}} \mathrel{+}= t_j^{\text{exe}}$
9:          end for
10:         Compute the execution time $t_i^{\text{sv}}$ of the service vehicle based on Equation (7)
11:         if $t_i^{\text{sv}} \le T^{\text{con}}_{S_i,S_j}$ and $t_i^{\text{sv}} \le T_i^{\min}$ then
12:             $T_i^{\min} = t_i^{\text{sv}}$
13:         end if
14:     end for
15: end for
16: Return the service vehicle selection decision
```

B. Hybrid Differential Teaching Optimization-based Task Scheduling Algorithm

When the request vehicle is within the coverage range of an RSU, the task can be offloaded to the edge server. If the edge server's resources are overloaded, the task is forwarded to a less loaded edge server. At this point, the task execution location decision, offloading ratio, and resource allocation strategy are coupled with each other. Therefore, this paper decomposes the problem into two subproblems: the resource allocation subproblem and the task offloading subproblem, and solves them iteratively.

1) Resource Allocation Subproblem: The resource allocation subproblem can be expressed as:

$$\min_{\delta_i, W_i, f_i^{RSU}} \sum_{i=1}^{N} \left(\frac{\delta_i d_i}{W_i r_i^{RSU}} + \frac{\delta_i d_i c_i}{f_i^{RSU}} + e_i \frac{d_i \delta_i}{R_{RSU}} \right) \tag{19}$$
$$s.t. \, (15c), (15d), (15e), (15f)$$

Where $r_i^{RSU} = \log_2 \left(1 + \frac{p_i |h|^2}{\mu_0 (d_{s_i,RSU})^{\vartheta}} \right)$ represents the channel capacity during the RSU transmission for each task.

By observation, the forwarding delay between servers is only related to the task offloading subproblem, so we assume that the task offloading location and offloading ratio are fixed. Thus, problem (19) can be written as:

$$\Gamma \left(W_i, f_i^{RSU} \right) = \sum_{i=1}^{N} \left(\frac{\delta_i d_i}{W_i r_i^{RSU}} + \frac{\delta_i d_i c_i}{f_i^{RSU}} \right) \tag{20}$$
$$s.t. \, (15c), (15d), (15e), (15f)$$

We can observe that constraints 15c - 15f are convex, so we only need to focus on the objective function. The Hessian matrix of problem (20) is as follows:

$$\begin{aligned} H &= \begin{bmatrix} \frac{\partial^2\Gamma}{\partial W_i \partial W_i} & \frac{\partial^2\Gamma}{\partial W_i \partial f_i^{RSU}} \\ \frac{\partial^2\Gamma}{\partial f_i^{RSU}\partial W_i} & \frac{\partial^2\Gamma}{\partial f_i^{RSU}\partial f_i^{RSU}} \end{bmatrix} \\ &= \frac{\partial^2\Gamma}{\partial W_i \partial W_i}\frac{\partial^2\Gamma}{\partial f_i^{RSU}\partial f_i^{RSU}} \\ &= \frac{2\delta_i d_i}{r_i^{RSU}(W_i)^3}\frac{2\delta_i d_i c_i}{\left(f_i^{RSU}\right)^3} > 0 \end{aligned} \tag{21}$$

Therefore, the Hessian matrix of problem (20) is positive definite, proving that this is a convex optimization problem. The Lagrangian function is defined as:

$$\begin{aligned} L\left(W_i, f_i^{RSU}, \lambda_1, \lambda_2\right) &= \sum_{i=1}^{N}\left(\frac{\delta_i d_i}{W_i r_i^{RSU}}\right. \\ &+ \left.\frac{\delta_i d_i c_i}{f_i^{RSU}}\right) + \lambda_1\left(\sum_{i=1}^{N} f_i^{RSU} - F_{RSU}\right) \\ &+ \lambda_2\left(\sum_{i=1}^{N} W_i - W_{RSU}\right) \end{aligned} \tag{22}$$

According to the Karush-Kuhn-Tucker (KKT) conditions, the allocated computational resources and bandwidth resources for each task in the edge server are given by:

$$\begin{aligned} f_i^{RSU} &= \frac{F_{RSU}\sqrt{\delta_i d_i c_i}}{\sum_{i=1}^{N}\sqrt{\delta_i d_i c_i}} \\ W_i &= \frac{W_{RSU}\sqrt{\delta_i d_i}}{\sum_{i=1}^{N}\sqrt{\delta_i d_i}} \end{aligned} \tag{23}$$

Thus, when the task offloading ratio and offloading location are fixed, the computational and bandwidth resources for task execution are allocated according to the formulas above.

2) Task Offloading Subproblem: Differential Evolution (DE) is a stochastic search-based global optimization algorithm that generates global search strategies through cooperation and competition among individuals within a population. It has been widely applied in the field of cloud-edge computing task scheduling [21]. The Differential Evolution Algorithm is used to solve the task execution location and offloading ratio search problem, which is then combined with the resource allocation subproblem. These two are alternately solved iteratively. However, traditional differential evolution algorithms often face slow convergence speeds when solving complex, high-dimensional, and multi-modal optimization problems. The Teaching Optimization Algorithm (TO) [22] simulates the learning process of teachers and students. It is a population-based random search algorithm characterized by simple concepts, fewer control parameters, and fast convergence speed, which can effectively compensate for the deficiencies of the differential evolution algorithm.

Therefore, this paper proposes a hybrid Differential Teaching Optimization (HDTO) task scheduling algorithm, which combines DE and TO algorithms by adding the teaching phase of the Teaching Optimization Algorithm into the differential evolution algorithm's iteration process, ensuring fast convergence. Furthermore, this method is applied to the task offloading decision-making process. Different mutation strategies can affect the performance of the DE algorithm in different ways. In the early stages of the algorithm, DE/Rand/1 has a wider search range, which is conducive to increasing the diversity of the population. In contrast, in the middle and later stages, DE/Best/1 searches more quickly, making it more suitable for fine local searches. To address this, an adaptive control strategy is used to improve the mutation strategy of the differential evolution algorithm by dynamically adjusting the weights to ensure the algorithm's execution efficiency. Additionally, to reduce the impact of randomly generated initial solutions on the algorithm's efficiency, a reverse learning strategy is introduced during the generation of the initial population, thus improving the convergence speed of the algorithm. The main idea of the algorithm is as follows:

Encoding: To apply the proposed algorithm to the task offloading subproblem, assume the population size is P, consisting of individuals G_p with N-dimensional decision variables. The individual $G_p = (g_{p,1}^i, g_{p,2}^i)$, where $p \in \{1, 2, ..., P\}$ and $i \in \{1, 2, ..., N\}$, represents the task offloading location and offloading ratio, respectively. In the task offloading subproblem, the current task offloading location and offloading ratio are determined, and based on the computed resources f_i^{RSU} and communication resources W_i from the resource allocation subproblem, the task execution time can be obtained. Therefore, the fitness function can be designed as:

$$\text{Fit}(G_p) = \sum_{i=1}^{N} t_i \tag{24}$$

Initialization: A reverse learning strategy is introduced to generate reverse solutions for the initial population using Equation (**??**). Then, based on Equation (26), new solutions are selected from both the initial and reverse solutions to form the initial population:

$$25G_p^{new} = u_b + l_b - \gamma_1 \times G_p \tag{25}$$

where G_p^{new} is the newly generated solution, u_b and l_b are the upper and lower bounds of the encoding values, and γ_1 is a random number between 0 and 1.

Next, the new solution is compared with the original solution:

$$G_p, i^{new'} = \begin{cases} G_p^{new}, & \text{if } \text{Fit}(G_p^{new}) < \text{Fit}(G_p) \\ G_p, & \text{otherwise} \end{cases} \tag{26}$$

Mutation: An adaptive weight factor ω is introduced to dynamically adjust the dominance of DE/Rand/1 and DE/Best/1 based on the iteration count, fully exploiting the advantages of both strategies at different stages of the algorithm:

$$\omega = \omega_{\text{min}} + (\omega_{\text{max}} - \omega_{\text{min}}) \times \frac{\text{Gen}_{\text{cur}}}{\text{Gen}_{\text{max}}} \tag{27}$$

$$\begin{aligned} G_p^{new_1} &= (1-\omega)\left(G_{p_1} + F \times (G_{p_2} - G_{p_3})\right) \\ &+ \omega\left(G_p + F \times (G_{\text{best}} - G_p)\right) \end{aligned} \tag{28}$$

where ω controls the mutation strategy weight, Gen_{max} is the maximum number of iterations, Gen_{cur} is the current iteration number, and F is the scaling factor. G_{p_1}, G_{p_2}, and G_{p_3} are randomly selected individuals.

Teaching: First, select the best individual in the current population as the teacher G_{teacher}, and then obtain the mean level of all individuals in the population G_{mean}. Using Equation (**??**), each individual undergoes a teaching phase to obtain the next generation:

$$29G_p^{new_2} = G_p^{new_1} + \gamma_2 \times (G_{\text{teacher}} - \gamma_3 \times G_{\text{mean}}) \qquad (29)$$

where γ_2 is a random number between 0 and 1, and γ_3 randomly takes values 1 or 2.

The differential evolution algorithm's mutation strategy is heavily influenced by individual solutions, neglecting the effect of global information on algorithm efficiency, which can result in slow convergence. However, the teaching phase of the Teaching Optimization Algorithm allows for the generation of the next generation based on the optimal individual and the mean level of the population, ensuring faster convergence. In the early stages of the algorithm, DE/Rand/1 dominates the mutation strategy, expanding the population diversity with a random search strategy, where individuals have significant differences, and each individual can acquire more knowledge based on the teaching phase. In the later stages of the algorithm, when individuals are more similar, the teaching optimization algorithm is less effective in acquiring new knowledge. At this point, DE/Best/1 dominates the mutation strategy, enhancing the focus of the search, thereby improving the algorithm's convergence precision.

Crossover: The differential evolution algorithm performs crossover operations on G_p^{new} and G_p to generate new solutions according to the binomial crossover formula (30):

$$G_{p,i}^{new} = \begin{cases} G_{p,i}^{new_2}, & \text{if } \gamma_4 < CR \\ G_{p,i}, & \text{otherwise} \end{cases} \qquad (30)$$

where γ_4 is a random number between 0 and 1, and CR is the crossover probability, with $i \in \{1, 2, \dots, N\}$.

Selection: After the above operations, the new generation of individuals is selected according to Equation (26). If the fitness value of the new individual is less than that of the original individual, the new individual is retained; otherwise, the original individual enters the next generation.

The Hybrid Differential Teaching Optimization (HDTO) task scheduling algorithm is shown in Algorithm 2.

Algorithm 2 HDTO Task Scheduling Algorithm Based on Hybrid Differential Teaching Optimization

```
1:  Initialize: population $P$, maximum iterations $I_{\max}$, weight factors $\omega_{\max}$, $\omega_{\min}$, scaling factor $F$, crossover probability $CR$
2:  Generate reverse solutions based on Equation (25) and reinitialize population $P$ based on Equation (26)
3:  while $I_{\text{cur}} \le I_{\max}$ do
4:    for $G_p$ in $P$ do
5:      Obtain the weight factor $\omega$ based on Equation (27), and perform mutation strategy based on Equation (28)
6:      Select the best individual in the population as the teacher $G_{\text{teacher}}$, and calculate the population mean $G_{\text{mean}}$
7:      Execute the teaching phase based on Equation (29)
8:      for $i$ in $N$ do
9:        if $\text{rand}(0,1) < CR$ then
10:         $G_{p,i}^{\text{new}} = G_{p,i}^{\text{new}_2}$
11:       end if
12:     end for
13:     Solve the resource allocation subproblem to obtain computing and bandwidth resource allocation strategies
14:     if $\text{Fit}(G_p^{\text{new}}) < \text{Fit}(G_p)$ then
15:       $G_p = G_p^{\text{new}}$
16:     end if
17:   end for
18:   $I_{\text{cur}} = I_{\text{cur}} + 1$
19: end while
20: Return the optimal task offloading strategy
```

C. Algorithm Complexity Analysis

The adaptive task offloading strategy in the vehicle-edge collaboration environment mainly consists of two parts: the best service vehicle selection algorithm and the task scheduling algorithm based on hybrid differential teaching optimization. Let the number of vehicles on the road be N, and the number of tasks waiting for service vehicles be Q. Then, the total time complexity of Algorithm 1 is $O(N^2Q)$. In the resource allocation subproblem, we only need to traverse all tasks once, so the time complexity of the resource allocation subproblem is $O(N)$. The hybrid differential teaching optimization algorithm mainly consists of the initialization phase, mutation phase, teaching phase, and selection phase, where the resource allocation strategy is applied in both the initialization and selection phases. Let the population size be P and the maximum number of iterations be I. Since the number of edge servers is relatively small, it can be neglected. The time complexity of each phase is as follows: In the initialization phase, the initial population solution is obtained through reverse learning strategy, and the total time complexity is $O(PN)$. The time complexities of the mutation phase and teaching phase are both $O(PN)$. The time complexity of the selection phase is $O(PN)$. Therefore, the time complexity of Algorithm 2 is $O(IPN)$. Thus, the overall time complexity is $O(IPN + N^2Q)$.

V. Experimental Results and Analysis

A. Experimental Setup

In this paper, a road topology consisting of a 2 km long two-way lane is simulated. Three roadside units (RSUs) carrying edge servers are evenly distributed along the roadside, with a distance of 500 m between adjacent RSUs. The coverage range of each RSU is 200 m, and vehicle speeds are randomly selected from the NGSIM highway dataset. Each vehicle's communication range is 100 m. Other experimental parameters are listed in Table 1.

To verify the effectiveness of the proposed Best Service Vehicle Selection Algorithm (VSA) and the Hybrid Differential Teaching Optimization Task Scheduling Algorithm (HDTO), comparisons are made with the following methods under the same conditions:

1) DE+VSA: The original Differential Evolution (DE) algorithm combined with the Best Service Vehicle Selection Algorithm (VSA) proposed in this paper.
2) HDTO+MR: The task offloading problem uses the Hybrid Differential Teaching Optimization Task Scheduling Algorithm (HDTO), while the vehicle selection uses a greedy algorithm to choose the service vehicle with the most remaining resources (Maximum Resources).
3) PSO+KM: Vehicles are clustered based on their position, speed, and direction. A greedy algorithm is used to select service vehicles within each cluster, and Particle Swarm Optimization (PSO) is applied to solve the task allocation and scheduling problem.
4) HDTO+VSA: The proposed Hybrid Differential Teaching Optimization Task Scheduling Algorithm (HDTO) combined with the Best Service Vehicle Selection Algorithm (VSA), which considers the collaboration between vehicles and edge servers to adaptively choose task offloading decisions.

TABLE I
EXPERIMENTAL PARAMETER SETTINGS (PART 1)

Parameter	Value
Number of Vehicles N	50-200
Channel Fading Coefficient	4.2
Edge Server Wireless Signal Coverage Radius (m)	200 m
Channel Background Noise Power (dBm)	-110
Vehicle Upload Power (dBm)	30
RSU Communication Bandwidth (MHz)	100

TABLE II
EXPERIMENTAL PARAMETER SETTINGS (PART 2)

Parameter	Value
Communication Bandwidth between Vehicles (MHz)	10
Vehicle Processor Frequency (GHz)	[1, 2]
Edge Server Single-Core Frequency (GHz)	5.0
Wired Link Bandwidth between Edge Servers (Mbps)	500
Task Input Data Size (MB)	[0.5, 1]
Task Required Computation Size (cycles/bit)	[500, 1000]

B. Experimental Results and Analysis

Figure 3 shows the fitness function values of different algorithms under the same number of iterations. Comparing the basic Differential Evolution (DE) algorithm with the Particle Swarm Optimization (PSO) algorithm, we can observe that in the early stages of execution, the HDTO algorithm converges faster. The original Differential Evolution algorithm has a strong search ability in the early stages but exhibits lower convergence. The initialization of the population using the reverse learning strategy in this paper helps expand the search range and enhance population diversity. The teaching mechanism ensures that the algorithm converges quickly. Therefore, in the early stages of the algorithm, it can quickly converge to a relatively optimal level. In the later stages of the algorithm's execution, the adaptive control strategy allows the algorithm

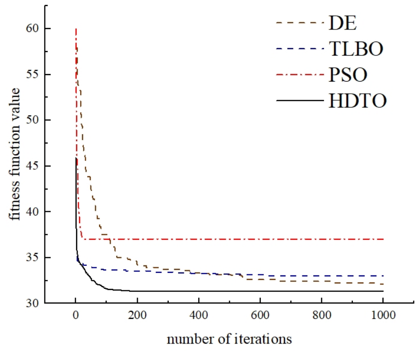


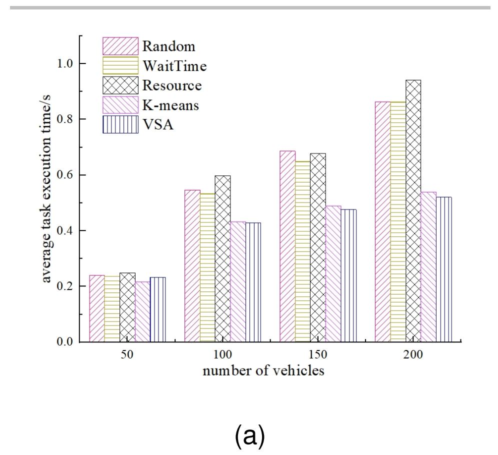
Fig. 3. Task execution waiting queue on service vehicle.

to search for the optimal solution as much as possible while maintaining stability.

Figure 4a shows the impact of different vehicle selection algorithms on the average task execution time. The comparison algorithms are: Random, which randomly selects service vehicles; WaitTime, which selects the vehicle with the shortest queue time; Resource, which selects the vehicle with the most resources; K-means, which clusters vehicles based on location and selects service vehicles from these clusters; and VSA, the optimal service vehicle selection algorithm proposed in this paper. As can be seen, the first three algorithms do not consider the impact of communication interruptions, so when task transmission fails, additional execution delays are incurred. Although K-means considers communication interruptions when selecting vehicles, the clustering process leads to an uneven distribution of service vehicles, and some clusters have insufficient resources. The optimal service vehicle selection algorithm proposed in this paper searches for all available nodes around itself, minimizing task execution time while ensuring communication stability, thus outperforming the other algorithms.

Figure 4b shows the average task execution time under different vehicle quantities in the vehicle-edge collaboration scenario. When the number of vehicles is small, the difference between algorithms is not significant, as computational resources are sufficient, and each request can be assigned enough resources for task offloading. However, as the number of vehicles increases, the HDTO algorithm gradually outperforms the other algorithms. The HDTO+MR algorithm, which uses a greedy approach to select the service vehicle with the most resources, leads to inefficient resource utilization, significantly increasing task delay. The HDTO algorithm, on the other hand, maximizes the utilization of both service vehicle and edge server resources, ensuring good execution efficiency even when the number of vehicles and tasks is large.

Figure 5 shows the computational load of each edge server when different algorithms are applied. When the computational capabilities of each edge server are the same, it can be seen that the HDTO algorithm proposed in this paper effectively balances the load among the edge servers. The HDTO algorithm, which incorporates a reverse learning strategy to initialize the population, expands the search space, enhances



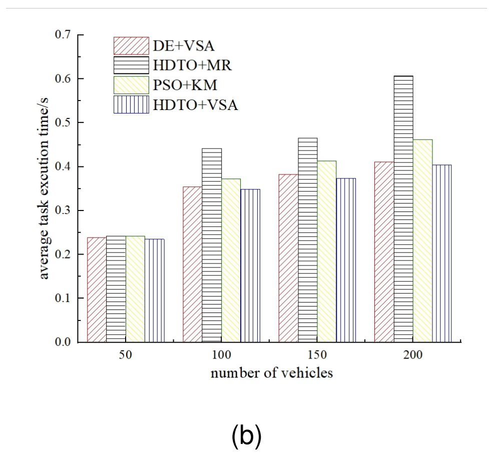


Fig. 4. The impact of different vehicle selection algorithms and the number of users under the vehicle-edge collaboration scenario. (a) average task execution time of service vehicles. (b) average task execution time under the vehicle-edge collaboration scenario.

population diversity, and, in the later stages of algorithm execution, ensures algorithm stability through an adaptive control strategy, while also striving to find the optimal solution. Therefore, the HDTO algorithm can more effectively coordinate task scheduling among edge servers, ensuring that the computational load of each edge server is similar, balancing the workload of the edge servers, and reducing task execution delay.

Figure 6 shows the average task execution time of each algorithm under different task sizes. LOC represents the time taken for the task to be executed locally. It can be seen that HDTO+MR and PSO+KM have much higher delays than the HDTO+VSA algorithm. This is because the first two algorithms use greedy strategies when selecting service vehicles: one selects the vehicle with the most resources, and the other selects the vehicle with the most stable communication. In contrast, HDTO+VSA considers both vehicle and edge server collaboration and adaptively selects the optimal offloading decision, avoiding blind vehicle selection for task offloading. As a result, even when the number of tasks increases, it still ensures lower task execution delays.

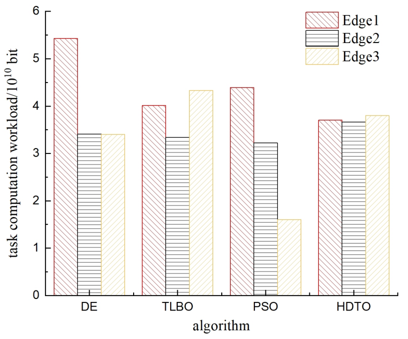


Fig. 5. Computational load of different types of edge servers under different algorithms.

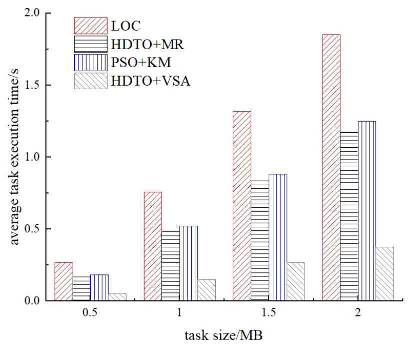


Fig. 6. average task execution delay under different task sizes.

VI. Conclusion

This paper addresses the issues of delayed task offloading and long task completion delays caused by limited communication coverage between vehicles and edge servers, as well as limited computational resources in a vehicle-edge computing environment. An adaptive task offloading strategy is proposed in a vehicle-edge collaboration environment to obtain the optimal task scheduling decision, thereby improving task offloading efficiency and reducing task completion time. First, based on the vehicle locations and the communication range of edge servers, and considering the collaborative relationships between vehicles and edge servers, a task offloading model in the vehicle-edge collaboration environment is constructed with the goal of minimizing task completion time. Second, an adaptive task offloading strategy is designed based on vehicle locations. When a vehicle is outside the coverage of an edge server, an optimal service vehicle selection algorithm (VSA) is proposed, offloading tasks to the best service vehicle through collaboration among vehicles. When the vehicle is within the coverage of an edge server, a hybrid differential teaching optimization-based task scheduling algorithm (HDTO) is proposed, offloading tasks to the optimal edge server through collaboration among edge servers. Experimental results show that the proposed adaptive task offloading strategy can effectively reduce task execution time in vehicular networks.

REFERENCES

[1] Y.-J. Ku, S. Baidya, and S. Dey, “Uncertainty-aware task offloading for multi-vehicle perception fusion over vehicular edge computing,” *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14 906–14 923, 2023.

[2] J. Liu, K. Xue, Q. Miao, S. Li, X. Cui, D. Wang, and K. Li, “Mcvco: Multi-mec cooperative vehicular computation offloading,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 813–826, 2024.

[3] Y. Li, B. Yang, H. Wu, Q. Han, C. Chen, and X. Guan, “Joint offloading decision and resource allocation for vehicular fog-edge computing networks: A contract-stackelberg approach,” *IEEE Internet of Things Journal*, vol. 9, no. 17, pp. 15 969–15 982, 2022.

[4] P. Dai, Y. Huang, K. Hu, X. Wu, H. Xing, and Z. Yu, “Meta reinforcement learning for multi-task offloading in vehicular edge computing,” *IEEE Transactions on Mobile Computing*, vol. 23, no. 3, pp. 2123–2138, 2024.

[5] A. Akbar and S. B. Belhaouarie, “Save: Self-aware vehicular edge computing with efficient resource allocation,” in *2023 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2023, pp. 157–162.

[6] D. F. H. B. e. a. Liu, G., “A collaborative computation and dependency-aware task offloading method for vehicular edge computing: a reinforcement learning approach,” *Cloud Comp*, vol. 23, no. 3, pp. 2123–2138, 2022.

[7] Z. J. e. a. ZHANG Degan, LI Xia, “New method of task offloading in mobile edge computing for vehicles based on simulated annealing mechanism,” *Journal of Electronics Information Technology*, vol. 44, no. 9, pp. 3220–3230, 2022.

[8] H. Zhu, Q. Wu, X.-J. Wu, Q. Fan, P. Fan, and J. Wang, “Decentralized power allocation for mimo-noma vehicular edge computing based on deep reinforcement learning,” *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 12 770–12 782, 2022.

[9] C. Huang, Q. Fu, C. Wang, and Z. Li, “Joint task offloading and scheduling algorithm in vehicular edge computing networks,” in *2023 IEEE 10th International Conference on Cyber Security and Cloud Computing (CSCloud)/2023 IEEE 9th International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2023, pp. 318–323.

[10] L. Qian, P. Sun, K. Yang, A. Boukerche, and L. Song, “A novel multi-factor aware online scheduling method for improving vehicular edge computing efficiency,” in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 3357–3362.

[11] M. D. Hossain, T. Sultana, M. A. Hossain, M. A. Layek, M. I. Hossain, P. P. Sone, G.-W. Lee, and E.-N. Huh, “Dynamic task offloading for cloud-assisted vehicular edge computing networks: A non-cooperative game theoretic approach,” *Sensors*, vol. 22, no. 10, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/10/3678

[12] C. Tang, C. Zhu, H. Wu, Q. Li, and J. J. P. C. Rodrigues, “Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing,” *IEEE Internet of Things Journal*, vol. 9, no. 7, pp. 5051–5064, 2022.

[13] C. H. e. a. ZHU Sifeng, SONG Zhaowei, “Multi-objective optimization offloading decision with cloud-side-end collaboration in smart transportation scenarios,” *Journal of Xidian University*, vol. 51, no. 3, pp. 63–75, 2024.

[14] Z. Z. e. a. ZHANG Chao, ZHAO Hui, “A task scheduling method for minimizing completion time in edge collaborative environment,” *Journal of Xidian University*, vol. 51, no. 4, pp. 114–127, 2024.

[15] Y. Ming and S. Leng, “Collaborative task offloading for vehicular edge computing,” in *2022 IEEE 22nd International Conference on Communication Technology (ICCT)*, 2022, pp. 798–803.

[16] X. Chen, H. Guo, and J. Liu, “Efficient and trusted task offloading in vehicular edge computing networks,” in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 5201–5206.

[17] M. Gong, Y. Yoo, and S. Ahn, “Vehicular cloud forming and task scheduling for energy-efficient cooperative computing,” *IEEE Access*, vol. 11, pp. 3858–3871, 2023.

[18] W. Li, N. Zhang, Q. Liu, W. Feng, R. Ning, and S. Lin, “Scalable modulation based computation offloading in vehicular edge computing system,” in *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, 2020, pp. 1–5.

[19] S. Pang, N. Wang, M. Wang, S. Qiao, X. Zhai, and N. N. Xiong, “A smart network resource management system for high mobility edge computing in 5g internet of vehicles,” *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3179–3191, 2021.

[20] X. Y. e. a. ZHANG Haibo, BIAN Xia, “Blockchain-assisted vehicle reputation management method for vanet,” *Journal of Xidian University*, vol. 49, no. 4, pp. 49–59, 2022.

[21] Y. Laili, X. Wang, L. Zhang, and L. Ren, “Dsac-configured differential evolution for cloud–edge–device collaborative task scheduling,” *IEEE Transactions on Industrial Informatics*, vol. 20, no. 2, pp. 1753–1763, 2024.

[22] H. Rao, H. Jia, D. Wu, C. Wen, S. Li, Q. Liu, and L. Abualigah, “A modified group teaching optimization algorithm for solving constrained engineering optimization problems,” *Mathematics*, vol. 10, no. 20, 2022. [Online]. Available: https://www.mdpi.com/2227-7390/10/20/3765