Journal of Computers Vol. 36 No. 2, April 2025, pp. 153-167
https://doi.org/10.63367/199115992025043602011

Collaborative Sensing Method for Intelligent Driving of New Energy Vehicles Based on Mobile Edge Computing

Xu Liu*

Tangshan Polytechnic University,
Tangshan City 063299, Hebei Province, China
lx521855@126.com

Received 29 March 2025; Revised 7 April 2025; Accepted 16 April 2025

Abstract. This paper addresses the current situation of huge amounts of data transmission and computing in the edge computing strategy of Internet of Vehicles. Firstly, an edge computing network model is constructed, which consists of two parts: vehicles and roadside units. Then, a communication model among multiple vehicles and a perception model among multiple vehicles are built. Based on these models, the motion state of the vehicle end, the operation state of the local computer, the transmission state of the wireless communication network, and the operation state of the edge server are combined into a Markov process, and the quality of the decision-making process is evaluated by reward values. This facilitates the solution of the optimal task offloading strategy through reinforcement learning theory in the subsequent text. In the process of solving the optimal parameters, a federated learning algorithm with joint auxiliary training and adaptive sparsity is used to ensure that important devices can continuously participate in multiple rounds of federated training and model solving. Finally, experiments are conducted to compare the algorithm in this paper with the previous method. The comparison shows that the offloading strategy in this paper can achieve smaller delays. By setting different vehicle scales, the improvement of the perception coverage area through the collaborative control among different vehicles is verified.

Keywords: cooperative perception, edge computing, Markov, federated learning algorithm

1 Introduction

With the rapid development of wireless communication technology and the significant increase in the number of new energy vehicles, vehicular networks has become a hot technology of mobile ad hoc network [1]. Around the on-board network [2], new energy vehicles equipped with wireless communication equipment can realize data transmission between driving vehicles, and can also communicate with roadside infrastructure, such as road end cameras and on-board routes of parked vehicles [3]. On the other hand, the driver's demand for real-time road information and entertainment information is increasing during driving and passengers' riding, and the application scenario of the Internet of vehicles is broader. The Internet of vehicles is a new network form of mobile ad hoc network applied in the field of road traffic. One of its important applications is to provide information services to users. However, the data information of network content service providers is generally stored in the remote cloud, while the on-board users are at the edge of the network, which will cause large network delay when requesting cloud data. When a large number of users request the content of cloud data at the same time, it will have a large computing burden on the cloud, which will lead to network congestion and affect the user experience in extreme cases. For these applications that provide media and entertainment information, the primary concern is the quality of service of users. In general, the content or services requested by most vehicles are locally relevant. Storing them on the edge side can make better use of network resources, so the network structure of mobile edge computing can just adapt to the current driving needs [4].

The future development direction of automobiles is assisted driving and autonomous driving. The realization of the above driving scenarios cannot be separated from the functional modules of the vehicle's environmental perception module, path planning and decision-making module, and motion control module. The environmental perception module is the foundation of path planning and decision-making and motion control. By obtaining

* Corresponding Author

precise, reliable, and comprehensive environmental perception information through computing, autonomous behavior decisions and safe driving behaviors of unmanned vehicles can be achieved, reducing road traffic accidents. Cooperative perception technology expands the perception range and improves the accuracy and reliability of target detection by establishing network communication with neighboring vehicles and roadside base stations to share perception information. However, with the increase in vehicle density, cooperative perception technology not only enhances the vehicle's perception capability but also brings a large amount of data redundancy and computational load. Although existing cooperative perception strategies have reduced information redundancy to a certain extent, they have ignored the key factor of target detectability. Moreover, these strategies have not fully considered the dynamic cooperative perception requirements in mixed traffic environments. Therefore, based on the current research on cooperative perception technology for unmanned vehicles, this paper studies the problem of cooperative perception information redundancy and cooperative perception information sharing strategies by leveraging neural networks, deep reinforcement learning, and other technologies, aiming to design a more efficient and reliable cooperative perception solution for unmanned vehicles [5].

The current deficiencies of cooperative perception in autonomous driving in terms of environmental perception and computational latency are analyzed and summarized as follows:

1) In terms of computational latency, first, cooperative perception relies on the fusion of data from multiple sensors (cameras, lidars, etc.) and vehicle-to-vehicle (V2V)/vehicle-to-everything (V2X) communication. However, the data collection frequencies and communication delays of different sensors vary, making time synchronization difficult and affecting real-time decision-making. Second, edge computing resources are limited. Although edge computing (such as roadside units, RSUs) can reduce the reliance on the cloud, the computing power of on-board computing units (ECUs) and edge nodes is limited, making it difficult to efficiently process high-precision perception models (such as 3D object detection), which may increase inference latency. Finally, the instability of communication networks, 5G/V2X communication may experience signal attenuation or congestion in complex environments (such as tunnels, urban canyons), leading to transmission delays of critical perception data (such as the position information of adjacent vehicles), affecting the timeliness of cooperative decision-making.

2) In terms of environmental perception, first, regarding occlusion and blind spots, single-vehicle sensors have limited field of view (such as being occluded by large vehicles). Although cooperative perception can partially compensate for blind spots through multi-vehicle data sharing, in sparse traffic scenarios (such as single-vehicle driving) or when communication is interrupted, it may still miss detecting key obstacles. Second, cross-modal perception inconsistency, the perception results of different vehicles or sensors may conflict due to differences in perspective and algorithms (such as cameras misjudging shadows as obstacles, while lidars do not detect them), requiring complex fusion strategies, increasing computational burden and uncertainty. Finally, insufficient adaptability to extreme environments, in adverse weather conditions (such as heavy rain, heavy snow) or strong light interference, sensor performance declines (such as sparse lidar point clouds, camera overexposure), and the robustness of cooperative perception still needs to be improved.

The work done in this paper on the integration of edge computing and cooperative perception during vehicle driving is as follows:

1) An edge computing network model was constructed, which consists of two parts: vehicles and roadside units. Then, a communication model among multiple vehicles and a perception model among multiple vehicles were built;
2) The adaptive sparse federated learning algorithm was used to solve the Markov process, and the algorithm design process was completed;
3) A simulation experiment platform was built, and the effectiveness of the method proposed in this paper was verified through comparative experiments.

2 Related Work

Drawing on a large number of research results and summarizing the current achievements in edge computing and collaborative perception provides a more scientific approach for the overall framework of this article. Both domestic and foreign scholars have achieved corresponding results in edge computing and collaborative perception. In edge computing, this article mainly summarizes the achievements of edge computing in new energy vehicles.

Wenwang Li from the National University of defense technology, the research scheme is to unload the edge computing tasks to the edge server in the roadside unit. At the same time, the problem of resource preemption and task processing delay will be considered when multiple vehicles send unloading requests at the same time. From the perspective of multi-objective optimization, the delay and energy consumption of unloading are analyzed and calculated in detail to minimize the delay and cost, and a task unloading algorithm based on improved non dominated sorting genetic algorithm is proposed. Finally, by reasonably balancing the delay and energy consumption, the algorithm can further improve the performance and efficiency of the Internet of vehicles system and provide users with a safer and more convenient travel experience [6]. This approach demonstrates the potential of edge computing in reducing latency and optimizing resources in real-time vehicular networks.

Bin Qiu from Guilin University of Technology proposed a computing offloading and power allocation method based on deep reinforcement learning for the problems of vehicle mobility-induced time-varying channels and random task arrivals in the vehicular edge computing environment. The offloading model adopted a three-layer “end-edge-cloud” offloading model based on non-orthogonal multiple access in a two-lane scenario. Considering the communication, computing, and caching resources of this model as well as the mobility of vehicles, a joint optimization problem was further established to minimize the long-term cumulative total cost of power and caching delay for vehicular users. Finally, considering the dynamic, time-varying, and random characteristics of vehicular edge computing networks, a distributed intelligent algorithm based on deep deterministic policy gradient was proposed to obtain the optimal power allocation mechanism [7].

Hongwei Zhao from Shenyang University proposed a vehicle-road-side collaborative scenario model in response to the issues and challenges of vehicle edge computing. Taking vehicle density as the entry point, he defined the problem of minimizing the probability of communication link interruption and established a communication rate model based on traffic density. By integrating three strategies - vehicle offloading, pricing, and resource allocation - the system optimization objective was described as minimizing the cost on the vehicle side while maximizing the utility value on the RSU side. The idea of problem decomposition was introduced to reduce the coupling degree of the problem, converting the original optimization objective into a balance problem between offloading and pricing as well as a resource allocation problem. The existence of the Nash equilibrium point in the offloading and pricing game was verified, and a distributed algorithm based on Stackelberg game (SDA) was proposed to solve the optimization problem. Finally, simulation experiments verified the impact of traffic density on transmission rate and demonstrated that SDA reduced the offloading cost of vehicles by 24% and increased the revenue of RSUs by 11% [8].

In terms of vehicle perception collaboration, Daxin Tian from Beihang University developed a two-dimensional robust control method for mixed platoons in vehicle wireless networks. This method involves a leading CAV and multiple following HDVs, integrating robust information sensing and platoon control. To effectively detect and locate unknown obstacles in front of the leading CAV, a collaborative vehicle-infrastructure sensing scheme was proposed and integrated with the adaptive model predictive control scheme of the leading CAV. Finally, extensive simulations were conducted to verify our method. The simulation results show that the proposed method can filter out the misleading perception information from malicious attackers, significantly reduce the mean square error of obstacle perception, and approach the theoretical error lower bound [9].

Jin Zheng from Beihang University, in response to the fact that the accuracy of existing 3D object detection algorithms based on pseudo point clouds is far lower than that of those based on real LiDAR (Light Detection and Ranging) point clouds, has studied pseudo point cloud reconstruction and proposed a 3D object detection network suitable for pseudo point clouds. Considering that the pseudo point clouds obtained from image depth conversion are dense and gradually sparse with increasing depth, a depth-related pseudo point cloud sparsification method has been proposed, which reduces the subsequent computational load while retaining more effective pseudo point clouds at medium and long distances, achieving pseudo point cloud reconstruction. This paper proposes a 3D object detection network guided by LiDAR point clouds for feature distribution convergence and semantic association. During network training, a LiDAR point cloud branch is introduced to guide the generation of pseudo point cloud object features, making the generated pseudo point cloud feature distribution converge to that of LiDAR point clouds, thereby reducing the detection performance loss caused by inconsistent data sources. Experimental results show that the accuracy of 3D object detection based on pseudo point clouds has been improved by 0.57% [10]. This contribution enhances the feasibility of using pseudo point clouds in 3D object detection, a critical aspect of autonomous vehicle perception.

Bing Zhu from Jilin University, in response to the issue that abnormal vehicles and malicious attack information in cooperative networks can affect the authenticity and effectiveness of cooperative perception results, proposed a dynamic aggregation evaluation method for the trustworthiness of intelligent connected vehicles based on occlusion state discrimination and detection effectiveness recognition strategies, in combination with the detection results in vehicle cooperative networks. Simulation results show that the cooperative perception trust evaluation method proposed in this paper enhances the credibility of the detection results for cooperative perception vehicles, strengthens the robustness of the cooperative perception process of intelligent connected vehicles, and realizes the dynamic identification of the trust management model in the face of sudden malicious attacks from high-trust vehicles [11].

The main content of this article is developed based on the overall framework of mobile edge computing, enhancing the vehicle's obstacle avoidance and environmental adaptability through the collaborative perception of the vehicle end and the road end. This collaborative perception framework not only improves the safety of autonomous driving but also optimizes resource usage in vehicular networks, contributing to the advancement of intelligent transportation systems.

3 The Construction of the Edge Computing Network System Architecture

The Internet of Vehicles is a highly complex network environment with huge amounts of data transmission and computing. It requires efficient data processing and management methods. In a network composed of multiple mobile edge computing nodes, users need to comprehensively consider factors such as the idle status of server resources and transmission distance, and determine the optimal mobile edge computing node for data offloading by balancing the weights of various parameters [12].

Mobile Edge Computing (MEC) architecture is a technical framework that shifts computing power from traditional centralized cloud to the network edge, aiming to meet the demands of emerging applications by reducing data transmission latency, improving bandwidth utilization, and enhancing privacy protection. This architecture mainly consists of three layers: network layer, host layer, and system layer.

The network layer is the fundamental part of the MEC architecture, responsible for defining and managing the network composition of the entire system. This layer elaborates on the network structure within the MEC architecture, including but not limited to 3GPP networks, local networks, and external networks. Among them, the 3GPP network, as one of the core standards for mobile communication, provides MEC with seamless integration capabilities with cellular networks, enabling mobile devices to quickly access edge nodes and enjoy low-latency services.

The host layer is the core component of the MEC architecture. It abstracts physical hardware resources through virtualization technology, thereby supporting multiple virtual machine or container instances to run on the same physical device. This approach significantly improves resource utilization and provides isolated execution environments for different applications. Additionally, the host management system plays a crucial role in this layer, mainly including two functional modules: MEC platform management and virtualization infrastructure management.

The system layer is located at the top of the entire architecture and mainly focuses on the management of global resources and the behavioral norms of applications. This layer clarifies the management rules for available resources in the MEC architecture and stipulates the verification requirements for applications to ensure that all programs running on edge nodes comply with the established security and performance standards. To further enhance system efficiency, the system layer dynamically adjusts, allocates, and releases the virtualization infrastructure based on the specific needs of applications.

3.1 The Construction of An Edge Computing Network Framework

The multi-vehicle edge computing network architecture is a network architecture oriented to the Internet of Vehicles. The network framework is shown in Fig. 1. The network is composed of two parts: vehicles and roadside units. Vehicles represent mobile nodes in vehicles, which are usually equipped with various types of sensors and communication devices and can collect and transmit various types of data. Roadside units refer to edge computing devices placed on the roadside, typically including edge computing servers, base stations, and vehicles parked on the roadside carrying communication devices, etc. Roadside units have edge computing capabilities and data storage capabilities to support various application services in the Internet of Vehicles. In a multi-vehicle edge computing network, a specified length of traffic section is set, and this specified length of road section is divided into several distance intervals. A roadside unit is placed in each interval, and each roadside unit is a mobile edge computing node used to provide edge computing services and storage services [13].

Suppose that vehicles are traveling at a constant speed on the road, and during the offloading decision-making process, each vehicle will choose the nearest roadside unit for transmission and communication. Each vehicle user needs to handle a computationally intensive task. Vehicles and roadside units can communicate and collaborate with each other to achieve efficient and reliable operation of various application services. By leveraging the communication and computing resources between vehicles and roadside units, multi-vehicle edge computing networks can significantly enhance the efficiency and performance of the Internet of Vehicles.

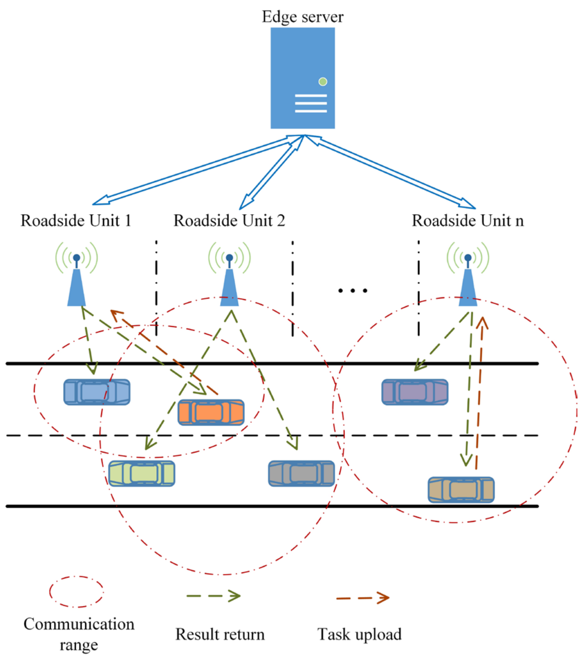

Fig. 1. Overall architecture of vehicle edge computing

The parameters included in the network architecture are shown in Table 1.

Table 1. Parameter list in network architecture

Parameter symbol	Parameter symbol description
$C = \{x \mid x = 1, 2, \cdots, n\}$	Vehicle matrix
$R = \{y \mid y = 1, 2, \cdots, m\}$	Roadside unit matrix
$T_x = (\alpha_x, \beta_x, \gamma_x)$	Intensive task matrix
α_x	Data volume of tasks
β_x	Task execution cycle
γ_x	Task delay
d_n	User offloading decision
d	System offloading decision
f_{local}	Local working frequency of device CPU
T_{cd}	Device computing latency
e_x	Energy consumption coefficient of local devices
η_t , η_e	Calculate the corresponding weights of computing delay and computing energy consumption.
$C_{S.power}$	The computing power of the server
n_{iy}	The number of users offloading computing tasks to the same edge computing server

3.2 The Construction of Communication Models

When a certain vehicle in the vehicle matrix, namely vehicle *x*, performs an offloading operation, the offloading decision is denoted as $d_x = d_x^m$, and the vehicle selects an edge computing server for computing offloading. In the Internet of Vehicles, due to the constantly changing positional relationship between vehicles and roadside units, this paper assumes that the arrival rate of vehicles in motion follows a Poisson distribution [14]. Therefore, the distance between vehicles and roadside units is constantly changing. The vehicle distance information is expressed as:

$$d = \sqrt{L^2 + \left(\frac{D}{2} - vt_i\right)^2} \tag{1}$$

Here, L represents the vertical distance between the roadside unit server and the horizontal road surface, D represents the coverage range of each roadside server, and t_i is the time it takes for the vehicle to pass through this roadside area. According to the Shannon formula, the information transmission rate between the vehicle and the roadside unit at a certain moment is expressed as:

$$V_{C\&R} = B_w \log_2\left(1 + \frac{p_c \cdot l_{path}^{-\delta} \cdot f_{channnel}^2}{\sigma^2}\right) \tag{2}$$

In the formula, $V_{C\&R}$ represents the information transmission rate between the vehicle and the roadside unit, B_w stands for the channel bandwidth during the wireless transmission process, P_c represents the transmission power of the vehicle equipment, $l_{path}^{-\delta}$ represents the path loss during the transmission between the vehicle and the roadside equipment, δ represents the path loss factor, $f_{channel}$ indicates the channel fading coefficient of the uplink, and σ^2 represents the power of Gaussian white noise. Then, the delay and energy generated by local computing at time *t* are expressed as:

$$T_{x.cd} = \frac{\beta_x}{f_{local}} \tag{3}$$

$$E_x = e_x \beta_x \tag{4}$$

The total cost of edge computing tasks executed locally is represented by the weighted sum of computing delay and computing energy consumption, as shown below:

$$C_x = \eta_t T_{x.cd} + \eta_e E_x \tag{5}$$

In the formula, the sum of the weights of the delay coefficient η_t and the energy coefficient η_e is 1. Assuming that all vehicles are treated fairly when allocating computing resources, when the vehicle's computing tasks are offloaded to the edge computing server for execution, the server's computing capacity $C_{S\cdot power}$ will be evenly distributed to the vehicle *x* based on the resource usage. The allocation process is expressed as follows:

$$C_{x.S.power} = \frac{C_{S.power}}{\sum_{i \in v, n_{iy} = n_{xy}} n_{iy}(t)} \tag{6}$$

Then the total cost of task offloading to the edge server is expressed as:

$$C_{total} = \eta_t \frac{\beta_n}{C_{x.S.power}} + \eta_e \frac{\alpha_n}{V_{C\&R}} \tag{7}$$

3.3 The Network Model of Edge Computing

Combining the delay and energy consumption formulas provided in the previous text, the mathematical expression for the total cost of computing offloading in the vehicle edge computing system can be obtained as:

$$\min C_{total} = \eta_t \sum_{x \in C} \left(d_x^0 T_{x.cd}(t) + d_x^m T_{r.cd}(t) \right) + \eta_e \sum_{x \in C} \left(d_x^0 E_x(t) + dE_r(t) \right) \tag{8}$$

When $d_x = d_x^0$ occurs, the vehicle selects the edge computing server for computing offloading. When $d_x = d_x^m$ happens, the vehicle user opts to offload the task to the nearest edge computing server.

3.4 Establishment of Multi-Vehicle Perception Model

In the process of multi-vehicle cooperative control in edge computing, each vehicle not only relies on its own on-board sensor system to collect environmental data, but also acquires additional environmental information through communication with cooperative vehicles [15]. This information sharing mechanism significantly expands the perception range and accuracy of environmental perception of individual vehicles. As mentioned in Section 3.3, the vehicle in this paper is denoted as $C = \{x | \ x = 1, 2, ..., n\}$. When vehicles are cooperatively controlled, they can communicate with each other, and the control objective is to achieve better detection of surrounding environmental targets during driving by sharing deep features among multiple vehicles. Since each driving vehicle is equipped with sensors necessary for autonomous driving, such as linear radar, laser scanning radar, and high-definition cameras, these devices can capture high-precision environmental data around the vehicle in real time. Through real-time data sharing among vehicles, a comprehensive perception of the surrounding environment is achieved.

For the i-th cooperative vehicle x_i, its raw sensing data and detection output are represented by $Data_i$ and Out_i respectively. Cooperative vehicle x_i uses an encoder to extract features $Feature_i$ from its raw sensing data $Data_i$. The extraction process requires effective processing of the sensor data obtained from the vehicle's lidar, high-definition cameras, and other sensors, thereby obtaining features that are rich in information and deeply compressed. The feature representation method is as follows:

$$Feature_i = Encoder(Data_i) \tag{9}$$

In the formula, *Encoder* represents the encoder extraction process. After the extraction is completed, the cooperative vehicle x_i fuses its own features with the features of the data obtained from other vehicles to form a fused feature $F_{feature}(i)$. Through time synchronization and spatial calibration, it ensures that the features from different vehicles can be effectively fused [16].

$$F_{feature}(i) = \cup_{x=1}^{n} \cdot Feature_i \tag{10}$$

Then, the collaborative fusion model F_{model} is used to process $F_{feature}(i)$, obtaining the fused feature M_i . This model is responsible for handling and optimizing the heterogeneous features transmitted from different driving vehicles to enhance the accuracy and reliability of the target detection results of the surrounding environment.

$$M_i = F_{\text{model}}\left(F_{feature}(i)\right) \tag{11}$$

Finally, the cooperative vehicle x_i uses the decoder *Decoder* to parse the detection results Out_i of the environmental detection targets from the fused features M_i .

$$Out_i = Decoder(M_i) \tag{12}$$

The environmental target detection result Out_i of the cooperative vehicle x_i is represented by a three-dimensional bounding box, and the representation method is as follows:

$$Out_i = \left(o_i^1, \cdots, o_i^K\right) \tag{13}$$

The bounding box contains the coordinate and shape information of the detected object, which is represented as follows:

$$o_i^k = \left(x, y, z, w, l, h, \alpha\right) \tag{14}$$

In the formula, (x, y, z) represents the center position of the object's bounding box, (w, l, h) represents the length, width and height of the bounding box, and α represents the angle indicating the tilt degree of the bounding box.

Based on the above-mentioned edge computing and collaborative perception model, a Markov process model is further established to solve the optimal resource allocation and target solution for edge computing and collaborative perception.

4 Modeling of Markov Decision Process in Edge Computing

Throughout the entire edge computing collaborative driving process, the motion state of the vehicle, the operation state of the local computer, the transmission state of the wireless communication network, and the operation state of the edge server can be modeled as a Markov process [17]. Therefore, the quality of the decision-making process can be evaluated by defining reward values, which provides convenience for further solving the optimal task offloading strategy based on the reinforcement learning theory [18].

Computational task offloading refers to the process of transferring local computing tasks from a local computer to an edge server for processing via wireless data transmission. Once the tasks are completed, the results are sent back to the local computer through wireless data transmission. In this study, the local computer specifically refers to the computing unit on board a vehicle or the computing unit of a roadside device. The decision-making process of task offloading involves an agent generating the decision results for task offloading based on a certain mapping relationship. Different task offloading strategies have a significant impact on the overall performance of a mobile edge computing system. Moreover, the design of task offloading strategies highly depends on the specific parameters of the edge computing system and the set optimization goals. For instance, when the wireless transmission rate is high and the edge server has sufficient computing power, if the goal is to optimize system latency, offloading as many tasks as possible to the edge server is a better strategy. However, for a system aiming to minimize energy consumption, since both data transmission and edge server operations bring additional energy costs, offloading more tasks to the edge server may not be the optimal choice [19].

In this context, the modeling process selects the vehicle coordinates and vehicle speed to represent the motion state at the vehicle end; describes the state of the computing task by calculating the data size of the task and the number of CPU cycles required per bit of data; and represents the execution state of the task by calculating the execution location of the task. Additionally, the aforementioned Markov process is also used to depict the computing states of the local computer and edge server. To simplify the modeling process, the following assumptions are made in this paper:

1) Within the wireless communication coverage range of each cellular base station, the wireless transmission state of data is approximately considered consistent and unchanged, not related to the vehicle's motion state, and the same cellular base station can simultaneously access multiple computing tasks;

2) In this mobile edge computing system, whether it is the local computer or the edge server, once a computing task is executed on the CPU, all computing power is released, that is, the scheduling of hardware resources only depends on the computing task offloading strategy.

4.1 State Space Representation

The entire mobile edge computing system consists of multiple cellular base stations. Taking the base station *y* of a certain cellular network as the center, for a specific computing task *j*, the system state can be expressed as:

$$s_{i,j} \triangleq \left\{ o_j^k, p_j, v_j, \varepsilon_j \right\} \tag{15}$$

In the formula, p_j represents the initial position coordinates generated by the computing task, v_j represents the movement speed of the vehicle, and ε_j represents the execution position of the computing task. Specifically, when $\varepsilon_j = 0$ occurs, it indicates that the task is executed on the local computer, and when $\varepsilon_j = 1$ occurs, it indicates that the task is executed on the edge server.

4.2 Representation of Action Space

Based on the modeling results of the system state space in the previous subsection, if the system action is to offload to the edge server, the task execution state ε_j will be set to 1, and if the system action is to arrange for execution on the local computer, ε_j will be set to 0. The expression of the action space is as follows:

$$a_{i,j} = \left\{ a_{i,j} \middle| a_{i,j} \in [0,1] \right\} \tag{16}$$

In the formula, $a_{i,j} = 0$ represents the execution of the computing task on the local computer, and $a_{i,j} = 1$ indicates that the computing task is offloaded to the edge server for execution.

4.3 The Reward Function Represents

After obtaining the cost function formula of a specific computing task Out_i in the system, the formula is expressed as follows:

$$O\left(T_{x.cd}(t), E_x\left(t\right)\right) = \beta T_{x.cd}(t) + (1-\beta) E_x\left(t\right) \tag{17}$$

For the entire system, considering the overall cost generated by the system at a certain moment to complete all the computing tasks within it, the cost for the action strategy that changes the system's state space from $s_{i,j}(n)$ to $s_{i,j}(n+1)$, with the involved task queue list represented as $\{o_j^1, o_j^2, ..., o_j^n\}$, the total cost value of the entire process is expressed as follows:

$$O\left(a_n, s_{i,j}\left(n\right)\right) = \sum\nolimits_{j=1}^{n} O\left(\sum T_{x.cd}(t), \sum E_x\left(t\right)\right) \tag{18}$$

To achieve the optimal performance of the system and minimize the system's cost during the task offloading process, the calculation results of the reward function and the cost function should be in a complementary relationship. The expression is as follows:

$$R\left(s_{i.j}\left(n\right), a_n, s_{i,j}\left(n+1\right)\right) \triangleq \varphi O\left(a_n, s_{i,j}\left(n\right)\right) \tag{18}$$

In the equation, A is the balance coefficient, set to a negative value. This coefficient is also used to balance the relative magnitudes of the reward function value and the cost value. Clearly, from the equation of the total system cost value, the cost value is a non-negative number, so the reward value will be a non-positive number. The objective of solving the optimal task offloading strategy is to maximize this non-positive number.

The above process completes the construction of the Markov model for the edge computing collaborative perception task in this paper. This paper uses the joint auxiliary training adaptive sparse federated learning algorithm to ensure that important devices can continuously participate in multiple rounds of federated training and model solving [20]. The algorithm comprehensively considers the computing power, energy, and federated model training process of each device, and adaptively adjusts the model sparsity within each communication round, dynamically deciding the auxiliary training devices and the allocation of auxiliary resources [21].

The pseudo-code of the algorithm is represented as follows:

```
Initialize global model parameters θ_global
Initialize local model parameters θ_local for each client
Initialize sparsity mask M for each client
Initialize learning rate η
Initialize number of communication rounds T
Initialize number of local epochs E
for t in range(T):
    selected_clients = select_clients(clients, fraction=C)
        for client in selected_clients:
         θ_local = θ_global
        for e in range(E):
         θ_local, M = adaptive_sparse_training(θ_local, M, client.data, η)
         upload_update(client, θ_local)
         θ_global = aggregate_updates(selected_clients)
        M_global = update_global_sparsity_mask(selected_clients)
        η = update_learning_rate(η, t)
return θ_global
def adaptive_sparse_training(θ_local, M, data, η):
    for batch in data:
               gradients = compute_gradients(θ_local, batch)
                  gradients = apply_sparsity_mask(gradients, M)
                  θ_local = θ_local - η * gradients
                  M = update_sparsity_mask(M, gradients)
        return θ_local, M
def update_sparsity_mask(M, gradients):
       M = apply_sparsity_criteria(M, gradients)
    return M
def aggregate_updates(selected_clients):
   θ_global = weighted_average([client.θ_local for client in selected_cli-
ents])
    return θ_global
def update_global_sparsity_mask(selected_clients):
      M_global = aggregate_sparsity_masks([client.M for client in select-
ed_clients])
    return M_global
def update_learning_rate(η, t):
    η = η * (1 / (1 + decay_rate * t))
    return η
```

The network parameters are updated using the deterministic policy gradient, which is effectively applied in the reinforcement learning decision-making of continuous action spaces. The overall network structure is shown in Fig. 2.

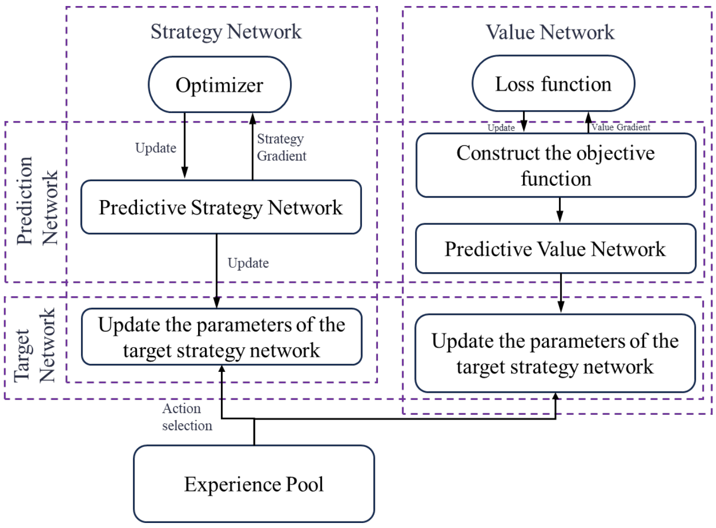

Fig. 2. Schematic diagram of algorithm structure

The optimization objective is $O\left(a_n, s_{i,j}(n)\right)$ and is defined as the cumulative discounted reward. The gradient formula of the value network sampling estimation is expressed as follows:

$$\nabla L\left(R\left(s_{i,j}(n)\right)\right) = E\left[\beta_x + \phi\left(s_{i,j}(n+1), \mu\left(s_{i,j}(n+1), \theta\right), R\left(s_{i,j}(n)\right)\right)\right] \tag{19}$$

In the formula, ϕ and μ represent the weights of the target policy network and the target value network respectively. By adding random noise to the action space, the interaction with the environment is increased, making the exploration of the policy more thorough. Meanwhile, the experience replay mechanism stores the training data and randomly samples it to update the model, reducing the correlation of sample data. Within multiple communication rounds of federated learning, fixing the model sparsity ratio is not conducive to the training of an accurate federated model. In the early stage of federated learning, the model parameters are far from the optimal values, and reducing the sparsity ratio at this time helps the model converge in the right direction; as the aggregation of multiple communication rounds progresses, the model parameters tend to stabilize, and increasing the sparsity ratio at this time helps reduce redundant communication and computation. Therefore, within each communication round, the central server aggregates the local models returned by the devices and edge servers and adaptively calculates the current global model's sparsity ratio. Performing a Hadamard product operation between the mask matrix and the global model yields the global sparse model. Then, the central server sends the global sparse model and the corresponding mask matrix to the devices and edge servers participating in the training for the next round of training. The gradients of the local models are calculated with the Hadamard [22] product of the mask matrix of this round, meaning that only the remaining weight parameters need to be updated in the next round of training.

5 Experimental Results and Analysis

This paper first conducts simulation experiments on different auxiliary training optimization algorithms. The code is implemented through Python 3.9.10 and runs on a computer equipped with an NVIDIA GeForce RTX 2060 GPU. The overall framework of vehicle-end collaborative perception is shown in Fig. 3.

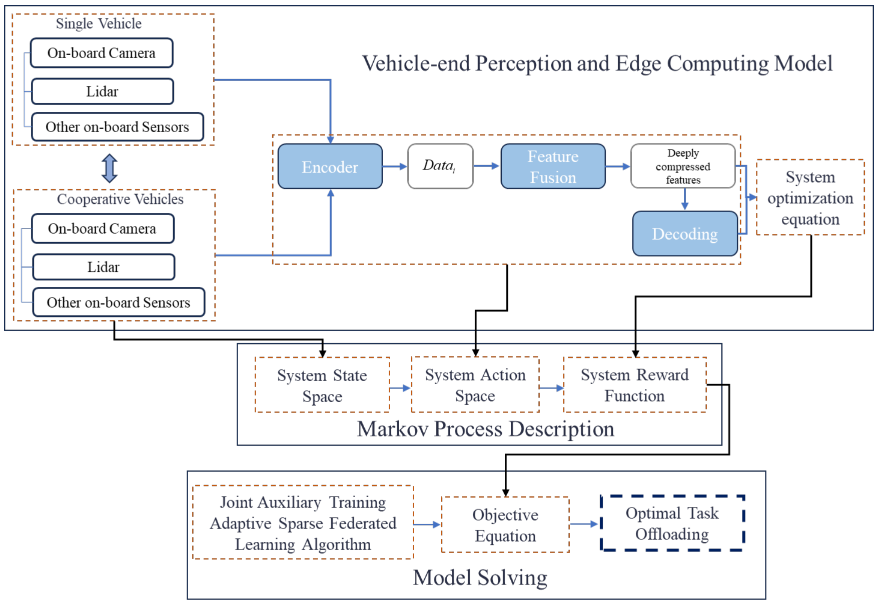

Fig. 3. Overall framework structure of vehicle-end perception

To compare the ability of the method proposed in this paper for edge computing task offloading, real simulation tasks were set up, which included object detection algorithms and roadside pedestrian positioning algorithms. As the training process of neural network parameters progresses, the cumulative reward value change curve of each sampled trajectory is shown in Fig. 4. As can be seen from the figure, the cumulative reward value increases sharply at the beginning of the training, indicating that the model performance is improving at a rapid speed. The reason for the early improvement is likely due to the optimization of edge computing task offloading, which makes calculation speed higher and processing more accurate. As training continues, the curve becomes more stable, indicating that the model has converged to a state where further significant improvements no longer occur on a regular basis. This can be the realization of diminishing returns of optimization when the system is nearing an almost optimal task offloading policy.

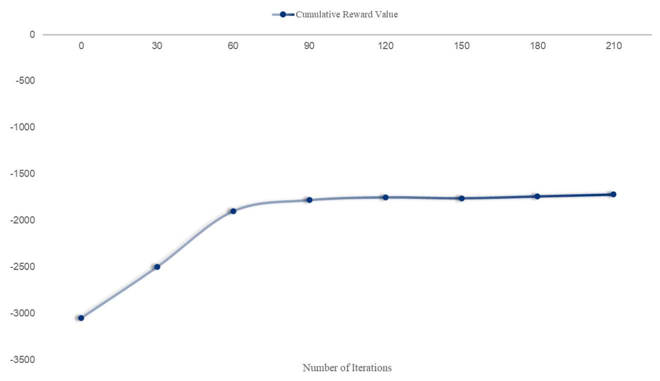

Fig. 4. The cumulative reward value change curve of the sampling trajectory

There was some jitter in the initial training stage. When the training process reached around the 75th generation, the curve tended to stabilize, and the parameters gradually updated to the optimal ones. The task offloading delay was compared for the tasks set in this paper using the strategy before and after improvement. The comparison results are shown in Fig. 5. As seen from the chart, the percentage of tasks with computation latency less than or equal to 20ms greatly improved after the improvement, improving from 12% prior to improvement to 54% following improvement. The percentage for latency between 20ms-40ms was 41% prior to improvement and improved slightly to 43% after improvement. However, for tasks with latency higher than 40ms, the percentage dropped drastically from 45% before improvement to only 5% after improvement. These results manifest the tremendous drop in the number of high-latency tasks, which indicates the accomplishment of the proposed method in minimizing task offloading latency and optimizing computation efficiency.

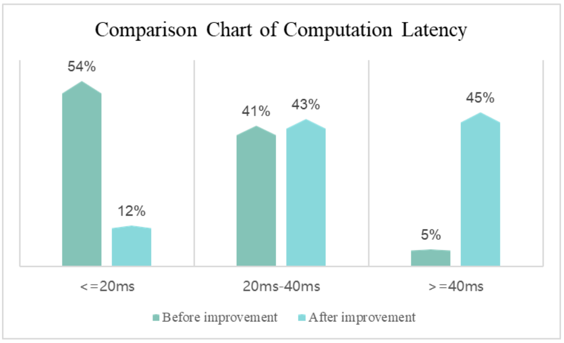

Fig. 5. Delay comparison effect

To verify the performance of cooperative perception, a MATLAB simulator is used to plan the driving scenarios of a fleet. The size of the fleet is set to 5 to 15 vehicles, and the trajectory of each vehicle is randomly planned. Therefore, the number of neighbors for each vehicle is dynamic. Moreover, in this section, the Mathematic simulation tool is adopted to simulate the perception coverage area of vehicles during dynamic driving. It enables the visualization of dynamic coverage areas. The communication coverage range of the leading vehicle is 100 meters. Thus, this section focuses on a four-lane road that is 1000 meters long and 20 meters wide. The fleet sizes are set to 5, 10, and 15 vehicles respectively.

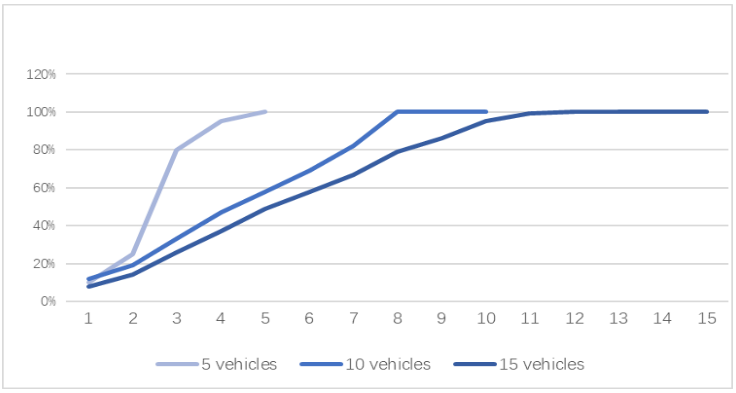

Fig. 6. Comparison of coverage areas of perception range

Multiple simulations were conducted for each scale to eliminate the error caused by a single experiment. The efficiency of the sensing coverage is shown in Fig. 6. As the number of selected vehicles increases, the cooperative sensing coverage area also enhances. When the number of vehicles reaches a certain quantity, it will cause a certain degree of redundant coverage. Since the sensing ranges of the first few vehicles do not overlap, as the number of selected vehicles increases, the overlapping situation also increases. When the vehicle scale is 5, this algorithm can ensure the maximum coverage area of cooperative sensing by considering the current vehicle's position and corresponding sensing capability, and giving priority to selecting vehicles with a larger sensing range to perform sensing operations. When the fleet scale increases to 15, the maximum sensing coverage area can be guaranteed when the number of selected vehicles reaches 12. This indicates that in the process of vehicle number selection, this algorithm takes into account the sensing overlap between vehicles, ensuring that the redundancy of sensing coverage is reduced during the vehicle selection process and improving the performance of cooperative sensing.

6 Conclusion

This paper designs a mobile edge computing method suitable for cooperative perception of intelligent driving vehicles, which can effectively expand the computing power and perception ability of intelligent driving vehicles. Through reinforcement learning methods, the optimal resource scheduling strategy of the cooperative control system is obtained, which saves the energy consumption of the system and optimizes the computing delay. A multi-vehicle cooperative perception fusion strategy is designed, which fuses the perception information of cameras and lidars, and describes the method of extracting target features through other vehicle-end networks. The final research method is verified in the simulation experiment scene. The simulation scene uses Matlab to build a multi-vehicle cooperative control environment. By comparing with the results of the control experiment group, it is shown that the vehicle-road cooperative perception strategy based on mobile edge computing proposed in this paper has significantly improved the perception range and computing performance of multi-vehicle perception tasks. Further research mainly includes the following two points: First, 5G communication equipment is in the process of large-scale commercial preparation, and the improvement effect of the communication network model under the 5G background can be further studied; second, the perception of actual intelligent vehicles is a large and complex process, and the vehicle-road cooperative perception of multiple targets needs to be further explored.

Firstly, 5G communication infrastructure is currently undergoing large-scale commercial deployment preparation. Further investigation into the performance enhancement of communication network models within the 5G framework is warranted. 5G technology facilitates highly efficient vehicle-to-everything (V2X) communication, enabling real-time transmission of high-precision sensor data such as LiDAR, camera, and millimeter-wave radar information. This capability effectively mitigates the limitations of single-vehicle perception in complex scenarios. Moreover, 5G edge computing supports rapid fusion and processing of distributed perception data, thereby improving the system's responsiveness to dynamic obstacles and emergency events. Additionally, the collaborative perception architecture for vehicular networking based on 5G networks promotes multi-vehicle data sharing, enabling the construction of a global perception model that enhances both the safety and robustness of autonomous driving systems. Future research should focus on optimizing 5G-enabled real-time perception algorithms, designing vehicle-cloud-edge collaborative computing frameworks, and developing novel communication-perception integrated network protocols, all of which are critical for supporting the large-scale deployment of advanced autonomous driving technologies.

Second, in terms of the lightweighting of the cooperative perception model, the lightweighting of the autonomous driving cooperative perception model is mainly achieved through multi-dimensional optimization. At the model level, lightweight network architectures (such as MobileNet and EfficientNet) and knowledge distillation techniques are adopted, reducing computational load through depthwise separable convolutions and channel attention mechanisms. At the data level, feature compression and sparsification are implemented, using autoencoders to compress key features and performing spatial/channel sparsification based on importance scores. Communication optimization involves adaptive bandwidth allocation and edge collaborative computing, offloading some tasks to roadside units to achieve joint optimization of computing and communication. In terms of hardware adaptation, 8/4-bit quantization, mixed precision, and hardware-specific optimizations are employed to significantly reduce storage and computing costs. Dynamic inference techniques utilize early exit mechanisms and input adaptive strategies to flexibly adjust computing resources based on scene complexity. These methods need to balance accuracy and efficiency, maintain critical safety redundancies, and consider the overall optimization effect of multi-vehicle collaboration. Current advanced solutions such as CoBEVT have demonstrated feasibility. In the future, by integrating neural architecture search technology, lightweighting will become more intelligent and efficient, laying the foundation for the large-scale deployment of autonomous driving.

References

[1] Q. Shi, L. Shan, T. Cheng, Q. Liu, C.-S. Wang, X. Zhang, Efficient Group Key Distribution Scheme Based on Quantum Random Numbers in VANETs, Automotive Engineering 46(2)(2024) 300-309+355.

[2] Z.-Z. Ma, Z.-Q. Sun, T.-F. Xue, Y.-B. Meng, J.-M. Zhang, H.-R. Tian, Design of data acquisition system for hybrid vehicle based on Internet of Things, Journal of Henan University of Engineering (Natural Science Edition) 35(2)(2023) 39-42.

[3] S.-R. Zhou, Y.-H. Lu, X.-W. Li, B.-Z. Fu, J.-D. Wang, X. Li, Pure camera-based bird's-eye-view perception in vehicle side and infrastructure side: a review, Journal of Image and Graphics 29(5)(2024) 1169-1187.

[4] X. Cheng, H.-T. Zhang, Z.-H. Yang, Z.-W. Huang, S.-J. Li, A.-L. Yu, Integrated sensing and communications for Internet of vehicles: current status and development trend, Journal on Communications 43(8)(2022) 188-202.

[5] B. Zhu, S.-Z. Jia, J. Zhao, J.-Y. Han, P.-X. Zhang, D.-J. Song, Review of Research on Decision-making and Planning for Automated Vehicles, China Journal of Highway and Transport 37(1)(2024) 215-240.

[6] W.-W. Li, H.-H. Zhou, S. Deng, W.-B. Ma, Y.-H. Wu, Joint Optimization of Delay and Energy Consumption of Tasks Offloading for Vehicular Edge Computing, Computer Science 51(11A)(2024) 658-664.

[7] B. Qiu, Y.-X. Wang, H.-L. Xiao, A Power Allocation Algorithm in Vehicular Edge Computing Networks Based on Deep Reinforcement Learning, Journal of Beijing University of Posts and Telecommunications 47(2)(2024) 81-89.

[8] H.-W. Zhao, J.-Y. You, Y.-Y. Wang, X.-K. Zhao, Dynamic Unloading Strategy of Vehicle Edge Computing Tasks Based on Traffic Density, Computer Science 50(S2)(2023) 730-736.

[9] D.-X. Tian, J.-S. Zhou, X. Han, P. Lang, Robust Platoon Control of Mixed Autonomous and Human-Driven Vehicles for Obstacle Collision Avoidance: A Cooperative Sensing-Based Adaptive Model Predictive Control Approach, Engineering 42(11)(2024) 244-266.

[10] J. Zheng, B.-T. Jiang, W. Peng, S. Wang, 3D Object Detection Based on Feature Distribution Convergence Guided by LiDar Point Cloud and Semantic Association, Acta Electronica Sinica 52(5)(2024) 1700-1715.

[11] B. Zhu, H.-Y. Jiang, J. Zhao, J.-Y. Han, Y.-C. Liu, A Method for Dynamically Calculating and Evaluating the Trustworthiness of Collaborative Perception of Intelligent Connected Vehicles, Automotive Engineering 45(8)(2023) 1383-1391+1407.

[12] G.-Q. Liu, X.-X. Zhang, H.-Y. Ma, H. Yan, Computational Task offloading Algorithms for Multi-edge Node Scenarios, Information and Control 52(5)(2023) 679-688.

[13] T. Wen, G.-Y. Li, Q. Gao, Application of Edge Computing of IoT in Real-time Data Analysis and Optimization of Intelligent Transportation, Application of IC 41(6)(2024) 228-229.

[14] Z.-H. Pei, L.-Y. Du, J. Ji, W. Chen, H.-J. Zheng, A Simulation of the Maximum Routing Hop Count of IoV Under Different Vehicle Densities, Journal of Transport Information and Safety 38(1)(2020) 92-99.

[15] H. Zhu, R.-F. Ni, A cooperative perception registration algorithm for intelligent and connected vehicles based on sparse semantic features, Chinese Journal of Scientific Instrument 44(10)(2023) 314-324.

[16] M.-R. Zhang, H. Yu, H. Lyu, L.-B. Jiang, L.-P. Li, L. Lu, Multimodal information fusion dynamic target recognition for autonomous driving, Journal of Chongqing University 47(4)(2024) 139-156.

[17] J.-K. Li, W.-W. Deng, B.-T. Ren, W.-Q. Wang, J. Ding, Automatic Driving Edge Scene Generation Method Based on Scene Dynamics and Reinforcement Learning, Automotive Engineering 44(7)(2022) 976-986.

[18] X.-F. Peng, A.-H. Liu, A survey of vehicular edge computing, Telecommunications Science 39(12)(2023) 19-28.

[19] H. Shen, L.-Q. Wang, Task Offloading Based on Mobile Edge Computing and Its Privacy-Preserving Issues: A Survey, Journal of Wuhan University(Natural Science Edition) 69(2)(2023) 258-269.

[20] Z.-H. Zhang, Q.-D. Li, Y. Fu, N.-X. He, T.-G. Gao, Adaptive Federated Deep Learning With Non-IID Data, Acta Automatica Sinica 49(12)(2023) 2493-2506.

[21] D.-B. Yan, H. Wen, W.-J. Hou, Y.-F. Wang, W.-D. Ma, F. Sun, Robust Multi-agent Federated Reinforcement Learning for Task Offloading, Communications Technology 57(8)(2024) 850-854.

[22] J.-J. Du, C. Qin, F.-W. Zou, X.-F. Li, X.-P. He, J.-Z. Feng, Inclusion Properties of Hadamard Product of Finitely Order Defined by Differential Subordination, Journal of Sichuan Normal University (Natural Science) 39(1)(2016) 71-75.