SAC-Based Collaborative Task Offloading and Path Optimization for Vehicular Edge Computing

Jun Wang, Lin Chai, Yumei Yang, and Wu Wang
School of Mathematics and Computer Science, Yunnan Minzu University, Kunming 650504, China

With the rapid development of Vehicle-to-Everything (V2X) and Vehicular Edge Computing (VEC), the massive computation-intensive tasks generated by intelligent vehicles during driving, including environmental perception, path planning, and autonomous driving decision-making, impose extremely high requirements on real-time performance and reliability. Traditional task offloading strategies often consider communication quality or computing resources in isolation, thereby neglecting the inherent coupling between a vehicle's dynamic driving path and its offloading decisions. This leads to low task completion rates, long travel times, and insufficient utilization of edge computing resources. Therefore, addressing the joint optimization problem in VEC scenarios, this paper proposes a deep reinforcement learning (DRL) algorithm based on an improved hybrid action space Soft Actor-Critic (SAC-Discrete). An agent capable of simultaneously handling continuous speed control and discrete offloading decisions is designed, enabling it to learn to collaboratively trade off task offloading and path optimization in dynamic traffic environments. This paper systematically compares the proposed algorithm with five baseline methods: greedy strategy, fixed path optimization, fixed offloading optimization, random strategy, and a discretized-action version of Vanilla SAC. Experimental results demonstrate that our algorithm converges rapidly and achieves high stability, and the proposed joint optimization framework significantly improves system efficiency and resource utilization while reducing average travel time.

***Index Terms*—VEC, Task Offloading, SAC-Discrete, DRL, Collaborative Optimization.**

I. Introduction

With the deep integration of fifth-generation mobile communication technology and the Internet of Things, V2X, as a core component of intelligent transportation systems, is undergoing revolutionary development. V2X technology constructs a real-time, reliable, and efficient intelligent transportation information interaction network by achieving comprehensive connectivity among Vehicle-to-Vehicle (V2V), Vehicle-to-Infrastructure (V2I), Vehicle-to-Pedestrian (V2P), and Vehicle-to-Network (V2N). In this context, emerging applications such as autonomous driving, high-definition map updates, real-time traffic analysis, and in-vehicle entertainment continue to emerge. These applications are often computation-intensive and delay-sensitive, posing unprecedented challenges to on-board computing resources [1].

Although traditional cloud computing models can provide powerful computing capabilities, network transmission delays and uncertainties due to the typical distance between cloud data centers and terminal devices make it difficult to meet the millisecond-level real-time requirements of V2X applications. VEC emerged as a solution, with its core idea being the sinking of computing, storage, and network resources to the network edge, closer to data sources and terminal devices. In VEC scenarios, edge servers deployed in Roadside Units (RSUs), base stations, or specific facilities can provide low-latency, high-bandwidth computing offloading services for moving vehicles, effectively alleviating the bottleneck of limited local vehicle computing resources [2].

However, the high mobility of vehicles, time-varying nature of wireless channels, and limited edge server resources make VEC task offloading an extremely complex dynamic optimization problem. During driving, vehicles not only need to decide "whether to offload" and "to whom to offload" computing tasks, but their own driving states such as speed, acceleration, and path selection also affect the channel quality with RSUs, thereby influencing the transmission success rate and delay of offloaded tasks [3].

Most current research treats path planning and task offloading as two independent optimization problems. This fragmented optimization paradigm cannot fully exploit the synergistic gains between them, potentially leading to overall system inefficiency. For example, vehicles may excessively detour to obtain better channel quality, significantly increasing travel time; or they may drive at high speeds to reach destinations, missing optimal communication and offloading opportunities [4]. Therefore, researching an intelligent algorithm capable of synergistically optimizing vehicle driving paths and computing task offloading decisions holds significant theoretical value and practical urgency for fully releasing the integration potential of VEC, and constructing truly efficient and reliable intelligent transportation systems.

Current research on VEC task offloading primarily focuses on resource allocation, offloading decisions, and computation scheduling, mainly employing optimization theory, game theory, and heuristic algorithms. When discussing optimization theory methods, researchers often formulate the problem as a Mixed Integer Nonlinear Programming problem, aiming to minimize total system delay or total energy consumption [5]. Such methods can theoretically obtain exact or approximate solutions, but typically suffer from high computational complexity, making them difficult to adapt to large-scale, highly dynamic real-time V2X environments. Game theory-based approaches treat vehicles as selfish decision-makers, analyzing competition and equilibrium in distributed offloading decisions through establishing non-cooperative game models [6]. These

Manuscript received January 16, 2026; revised April 1, 2026. Corresponding author: Wu Wang (email: wangwu@ymu.edu.cn).

methods can better characterize multi-agent interactions, but convergence analysis and Nash equilibrium solving often face difficulties. Heuristic rule-based methods, such as greedy strategies based on channel state [7], server load [8], or distance, are simple and easy to implement but lack long-term planning capability, easily falling into local optima and struggling to achieve global performance optimization.

In recent years, DRL has demonstrated significant advantages in solving such dynamic optimization problems due to its powerful sequential decision-making and environmental adaptability. Algorithm frameworks such as Deep Q-Network (DQN) [9], Policy Gradient (PG) [10], and Soft Actor-Critic (SAC) [11] have been introduced into the field of V2X resource management. However, existing research mostly designs action spaces as purely discrete or purely continuous, with few works systematically addressing the hybrid action space problem inherent in VEC joint optimization—the coexistence of "continuous path/speed control" and "discrete offloading target selection". Furthermore, existing RL methods often employ relatively simplistic reward function designs, failing to comprehensively and finely characterize the trade-offs among multiple objectives such as task completion quality, resource utilization efficiency, and traffic flow smoothness.

Addressing the aforementioned problems, this paper focuses on the joint optimization problem of vehicle driving paths and task offloading in VEC, aiming to design an algorithm capable of online learning and making intelligent collaborative decisions. The core research question of this paper is: Under dynamic traffic environments and wireless channel conditions, how can vehicles intelligently adjust their speed, select driving paths, and simultaneously decide when and to which edge server to offload computing tasks, to maximize task completion rate before deadlines while balancing travel time efficiency and edge computing resource utilization. The main contributions of this paper are as follows:

- Proposes an improved SAC-Discrete joint optimization algorithm: Targeting the characteristics of hybrid action space, designs a dual-head output policy network to separately handle the stochastic policy for continuous actions and the categorical policy for discrete actions, achieving effective learning of hybrid actions within a single framework.
- Constructs a refined collaborative decision state: In addition to conventional vehicle state, task state, and network state, innovatively introduces task type feature encoding, enabling the agent to recognize task size differences and providing basis for differentiated decision-making.
- Constructs a refined reward model: Creates a multi-objective composite reward function, containing not only task completion rewards but also rewards for utilizing RSU resources, driving efficiency, and global efficiency, guiding the agent to learn globally optimal collaborative strategies.
- Develops a SUMO-based simulation verification platform: Utilizes the open-source micro-traffic simulator SUMO to construct high-fidelity road scenarios and vehicle dynamics models, ensuring the authenticity of the research environment.

The remaining sections of this paper are organized as follows: Chapter 2 systematically reviews related work in VEC communication, offloading, and Reinforcement Learning (RL) in VEC, pointing out limitations of existing research. Chapter 3 elaborates on the system model studied in this paper, including scenario architecture, vehicle and communication, computing task models, and formally defines the Markov Decision Process (MDP) and problem formulation. Chapter 4 focuses on introducing the improved SAC-Discrete algorithm design, including network architecture, hybrid action processing mechanism, multi-objective reward function, and training process. Chapter 5 details the experimental setup, baseline methods, and evaluation metrics, and provides comprehensive analysis and discussion of experimental results. Chapter 6 summarizes the research work and conclusions of the paper.

II. Related Work

A. VEC Communication and Computing Offloading Research

VEC task offloading is an interdisciplinary research field involving wireless communication, mobile computing, resource management, and optimization theory. Early research primarily focused on theoretical modeling and analysis of the problem.

In terms of communication models, as Zhou et al. [12], researchers have widely adopted distance-based path loss models, small-scale fading models, and more complex V2I channel models to analyze V2X communication characteristics. For computing offloading scenarios, transmission delay is typically modeled as a function of task data size and instantaneous transmission rate, while transmission rate depends on signal-to-noise ratio, bandwidth, and modulation coding scheme, among other factors. Such research provides theoretical foundations for understanding the impact of channel dynamics on offloading performance, but often assumes fixed or simple vehicle trajectories, without fully considering the potential optimization space of vehicles actively adjusting paths to improve communication quality.

In terms of computing models, as in the work of Annu et al. [13] , tasks are typically abstracted as triplets consisting of data size, computational requirement, and deadline. RSU computing capability is modeled either as a fixed value or as a dynamic value based on current load. Queuing theory is commonly used to analyze task waiting time in server queues. Although these models can characterize computing resource contention, most treat the server processing procedure as a "black box," lacking differentiated handling considerations for heterogeneous task types—such as perception, decision-making, and entertainment tasks—which have different computational resource requirements.

Early optimization methods primarily employed mathematical tools such as convex optimization, integer programming, or mixed integer nonlinear programming, aiming to minimize total system delay, total energy consumption, or maximize user quality of service. For example, some studies transform stochastic optimization problems into deterministic optimization problems for each time slot using Lyapunov optimization frameworks [14]. Such methods are theoretically rigorous and

can provide performance bounds, but their high computational complexity and reliance on perfect channel state information make them difficult to cope with large-scale, highly dynamic real-time VEC environments. Furthermore, these methods typically require centralized controllers and global information, limiting scalability in distributed V2X scenarios.

B. Application of RL in V2X Resource Management

In recent years, RL has gained widespread attention in the field of V2X resource management due to its advantages in handling sequential decision problems and adapting to unknown environmental dynamics. The core idea is to enable agents (vehicles or controllers) to learn optimal policies through trial-and-error interaction with the environment, without requiring prior knowledge of the exact environment model.

DQN and its variants were among the earliest RL algorithms applied in this domain. Research has utilized them for base station selection, spectrum allocation, and simple binary offloading decisions. For example, Zhou et al. [15] uses DQN for vehicles to select the optimal RSU for task offloading, with state space containing vehicle position and channel state, and action space being the discrete set of RSUs. However, DQN algorithms are inherently designed for discrete action spaces and cannot directly handle continuous control variables such as vehicle speed and acceleration. Although discretization can handle continuous actions, this leads to dimensionality disaster and control precision loss.

PG and SAC offer possibilities for handling continuous action spaces. For example, Bi et al. [16] adopts Deep Deterministic Policy Gradient algorithms to jointly optimize vehicle transmission power and computing resource allocation. However, such methods face challenges when handling discrete decisions like "which RSU to choose". Existing SAC-based methods in V2X networks, such as Komeil et al. [17], predominantly simplify the problem by adopting purely discrete action spaces. However, this approach leads to limited action space resolution and introduces quantization errors.

To address competition and cooperation among vehicles, Multi-Agent Reinforcement Learning has also been introduced. Most work adopts a decentralized execution architecture, with each vehicle acting as an independent agent [18]. However, this introduces non-stationarity challenges, where each agent learns in an environment constantly changing due to other agents' learning. The introduction of the Centralized Training Decentralized Execution framework can effectively alleviate this problem, where the critic network can access global information during training, while during execution, each agent acts based solely on local observations.

Despite the preliminary success of RL in V2X, existing research still has shortcomings in addressing the core problem of joint optimization of task offloading and path planning. Most works either fix vehicle trajectories to optimize offloading [19] or optimize paths given offloading strategies [20], failing to place both in a unified, interactively learnable framework. Furthermore, reward function designs are often relatively simplistic (e.g., minimizing delay), lacking comprehensive consideration of multidimensional objectives such as task completion success rate, resource utilization fairness, and overall traffic flow efficiency.

Recently, several studies have explored advanced DRL techniques for VEC task offloading. More et al. [21] proposed DHPG (Deep Hybrid Policy Gradient), a novel DRL approach that operates in hybrid action spaces to jointly optimize service task placement and edge resource scaling in cellular V2N environments. Their work demonstrates the effectiveness of hybrid action space handling for VEC scenarios, though it focuses on resource scaling rather than path-offloading collaboration.

This paper's research work is precisely directed at the aforementioned challenges. We explicitly place vehicle continuous trajectory control and discrete task offloading decisions within a unified, MDP-based joint optimization framework. The core innovation of this paper lies in designing and implementing an algorithm based on improved SAC-Discrete, specifically targeting the hybrid action space characteristics in VEC joint optimization problems, integrating refined state representation and multi-objective reward functions, and achieving robust and accurate evaluation of algorithm performance in a high-fidelity SUMO simulation environment.

To better illustrate the distinct contributions of our work in the context of existing literature, Table I provides a comparative overview of representative methods for VEC task offloading and path optimization.

As summarized in Table I, many existing works treat path planning and task offloading as separate problems, using methods that are confined to either discrete or continuous action spaces. In contrast, our proposed SAC-Discrete algorithm uniquely addresses the hybrid action space inherent in VEC joint optimization. It explicitly models and learns the coupled relationship between vehicle path control (a continuous action) and offloading decisions (a discrete action) within a single, unified DRL framework. Furthermore, the use of the high-fidelity SUMO simulator for validation ensures the realism of our evaluations.

III. System Modeling and Optimization Problem Formalization

A. VEC System Architecture

The system architecture studied in this paper is shown in Fig. 1, depicting a typical urban road VEC scenario. This architecture contains three core layers: vehicle layer, edge computing layer, and control and optimization layer.

Specifically, the road network is a one-way main road with a length of 1000 meters, starting coordinates (0,0) and ending coordinates (1000,0). At 300 meters and 600 meters along the road, there are two intersections, each providing selectable branch paths that eventually merge and lead to the destination. Edge servers are two fixed RSUs with different computing capabilities. RSU_1 is located at coordinates (450, 300), RSU_2 at coordinates (450, -300). They connect to the edge network via wired links, possessing significantly more powerful computing capabilities than on-board units, but with limited coverage ranges. Vehicles are generated at the starting point following a Poisson process. Each vehicle carries a computing task upon generation, with task attributes including

TABLE I: Comparison of Our Method with Representative VEC Optimization Studies.

Methodology	Problem Focus	Action Space	Joint Optimization
DQN [16]	Offloading Decision	Discrete only	Path fixed, offloading only
DDPG [17]	Resource Allocation	Continuous only	Offloading fixed, resources only
Federated SAC [18]	Task Offloading	Discrete only	Path fixed, offloading only
Hyper-heuristic [20]	UAV-assisted Offloading	Discrete only	Trajectory fixed, offloading only
Heuristic [21]	Path Selection	Discrete only	Offloading fixed, path only
SAC-Discrete (Ours)	Offloading & Path	Hybrid (Continuous + Discrete)	Path & offloading jointly optimized

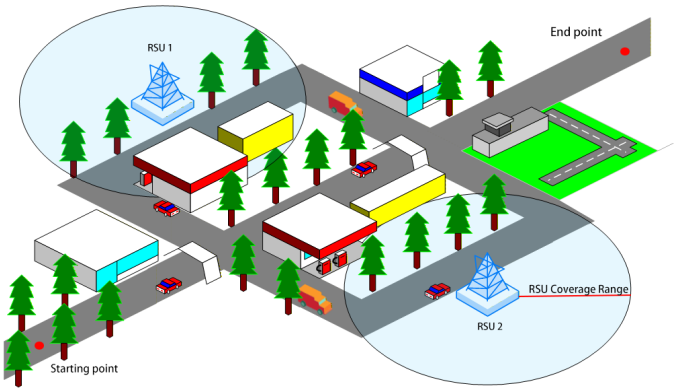

Fig. 1: VEC System Architecture.

size, computational requirement, and deadline randomly generated within reasonable ranges. Vehicles need to travel towards the end point while dynamically deciding on task execution methods.

Vehicles act as agents, with their decision cycles synchronized with the SUMO simulation step. At each decision time t, vehicles perform the following actions:

Perception: Vehicles acquire local environmental observations through on-board sensors and VEC communication, including their own state, relative positions to RSUs, estimated channel state, and limited global information broadcast from the edge controller, such as estimated RSU load.

Decision: Based on the current observation state S_t, the vehicle's reinforcement learning policy network π outputs a hybrid action:

$$A_t = (A_t^c, A_t^d), \tag{1}$$

where A_t^c is the continuous acceleration command, and A_t^d is the discrete offloading decision.

Execution: Vehicles adjust speed according to A_t^c and navigate towards different intersections based on the path selection algorithm. Simultaneously, task data is transmitted to the selected server or remains in the local queue based on A_t^d.

Feedback: The environment changes according to the vehicle's new position, task processing progress, etc., generating a new state s_{t+1} and providing the vehicle with an immediate reward r_t. The reward function comprehensively evaluates task progress, driving efficiency, and resource utilization.

B. Model Design

1) Dynamic Model of the Vehicle

Vehicle longitudinal kinematics are simulated by the Krauss car-following model within the SUMO simulator. For algorithm design convenience, we abstract this to a model partially controllable by the agent.

The state of vehicle i at time t is described by its position $x_{i,t}$, speed $v_{i,t}$, and acceleration $a_{i,t}$. Its dynamics are approximated as:

$$v_{i,t+1} = v_{i,t} + a_{i,t} \cdot \Delta t, \tag{2}$$

$$x_{i,t+1} = x_{i,t} + v_{i,t} \cdot \Delta t + \frac{1}{2} a_{i,t} \cdot (\Delta t)^2, \tag{3}$$

Where the actual operating acceleration $a_{i,t}$ must satisfy physical and traffic rule constraints: $a_{i,t} \in [a_{\min}, a_{\max}]$, and $v_{i,t+1} \in [0, v_{\max}]$.

Although path selection decisions are not directly output as actions, they are influenced by speed control and offloading decisions. When vehicles approach intersections, based on their current objectives, they intelligently choose to enter branch roads or stay on the main road. SUMO's routing logic combined with our custom decision logic implements this indirect path optimization.

2) Communication Model

The wireless channel quality between vehicle i and RSU_j ($j \in \{1, 2\}$) is crucial for determining offloading transmission delay. This paper adopts a channel gain model balancing simplicity and practicality. The channel power gain $h_{i,j,t}$ at time t is defined as:

$$h_{i,j,t} = \beta \cdot (d_{i,j,t})^{-\alpha} \cdot \boldsymbol{\zeta}_{i,j,t}, \tag{4}$$

Where $d_{i,j,t}$ is the Euclidean distance between vehicle i and RSU_j. α is the path loss exponent. β is a constant related to antenna gain and carrier frequency. $\zeta_{i,j,t}$ represents log-normal shadow fading, reflecting medium-scale signal fluctuations. Based on this, the received Signal-to-Noise Ratio $SNR_{i,j,t}$ can be defined as:

$$SNR_{i,j,t} = \frac{P_t \cdot h_{i,j,t}}{N_0 \cdot B}, \tag{5}$$

Where P_t is transmission power, N_0 is noise power spectral density, and B is bandwidth. Then, according to the Shannon–Hartley theorem, the instantaneous transmission rate $R_{i,j,t}$ is:

$$R_{i,j,t} = B \cdot \log_2 \left(1 + SNR_{i,j,t}\right), \tag{6}$$

Therefore, the theoretical transmission time required for a task of size D_i is :

$$\frac{D_i}{R_{i,j,t}}. \tag{7}$$

Within the RL framework, we do not precisely calculate transmission volume at each step; instead, we define a normalized channel quality indicator $CQ_{i,j,t} \in (0,1)$ as a state input directly reflecting transmission difficulty:

$$CQ_{i,j,t} = \frac{1}{1 + \frac{d_{i,j,t}}{R_c \cdot \kappa}}, \tag{8}$$

where κ is a scaling constant. The closer the vehicle is to the RSU, the closer CQ is to 1.

3) Computing Task and Resource Model

The task T_i generated by vehicle i is characterized by a triplet: $T_i = (D_i, C_i, \tau_i)$. In simulation, D_i follows a discrete distribution, including three task types: small, medium, and large. Computational requirement C_i represents the CPU cycles needed to complete the task. It is assumed that C_i is positively correlated with D_i to simulate tasks with different computational densities. τ_i is a random variable positively correlated with task complexity, representing the duration from task generation to mandatory completion, providing urgency signals for decision-making.

Each vehicle has a limited and fixed computational capability F_L. Each RSU_j possesses powerful computing capability $F_{E,j}$, $F_{E,j} \gg F_L$. RSU adopt a first-come-first-served approach to process offloaded tasks from multiple vehicles. Server load factor $L_{j,t} \in [0,1]$, defined as the ratio of current total queued computational demand to its maximum service capacity, is fed back to vehicles as part of the state to avoid offloading to overloaded servers.

4) MDP Modeling

We model each vehicle as an independent agent; within the multi-agent framework, the entire system can be viewed as a Partially Observable Markov Decision Process. The local observation state $s_{i,t}$ of vehicle i at time t is a 14-dimensional vector, specifically composed as follows:

Normalized position:

$$\frac{x_{i,t}}{1000}. \tag{9}$$

Normalized speed:

$$\frac{v_{i,t}}{v_{\max}}. \tag{10}$$

Normalized task data size:

$$\frac{D_i}{D_{\max}}. \tag{11}$$

Normalized remaining time:

$$\frac{\tau_i - (t - t_{i,gen})}{\tau_{\max}}. \tag{12}$$

where $t_{i,gen}$ is task generation time.

Normalized computational requirement:

$$\frac{C_{i,\text{ remaining}}}{C_{\max}}. \tag{13}$$

Task type encoding: a 3-dimensional one-hot vector, $[1,0,0]$ for small tasks, $[0,1,0]$ for medium tasks, $[0,0,1]$ for large tasks.

Channel quality: channel quality $CQ_{i,1,t}$ and $CQ_{i,2,t}$ for two RSU.

Server load: $L_{1,t}$ and $L_{2,t}$, representing the load factors of two RSU periodically obtained through V2I communication.

Traffic interaction information: normalized distance to preceding vehicle, and normalized relative speed. This helps the agent learn safe car-following behavior and avoid collisions.

The hybrid action space A_t specifically includes:

Continuous action A_t^c : acceleration $a_{i,t}$.

Discrete action A_t^d : offloading decision $a_{i,t} \in \{0,1,2\}$, representing local processing, offloading to RSU1 , and offloading to RSU2, respectively.

State transitions are dominated by the SUMO simulation environment, encapsulating vehicle dynamics, traffic flow interactions, task processing progress updates, and wireless channel variations. This is a complex, high-dimensional stochastic process, RL algorithms do not require knowledge of its exact form but learn through interactive sampling. The reward function R is the core for guiding agents to learn collaborative strategies. The immediate reward $r_{i,t}$ obtained by vehicle i at time t consists of the following components:

- Task Completion Reward R_{comp} : If the task is completed, a large positive reward will be given; otherwise, a large negative reward will be imposed.
- Progress Efficiency Reward R_{prog} : Encourages progress in task processing. Whenever a certain proportion of computation is completed, a small positive reward is given.
- RSU Usage Reward R_{rsu} : When selecting offloading action $o_{i,t} > 0$, a fixed small reward is given to encourage utilization of edge resources, but this reward is significantly smaller than the task completion reward to avoid resource abuse.
- Driving Efficiency Reward R_{speed} : Encourages vehicles to travel close to the road's expected speed v_{max} , penalizing excessively low speeds causing congestion or excessively high speeds leading to unsafety. Defined as negative squared error:

$$-\lambda_s \cdot \left(\boldsymbol{v}_{i,t} - \boldsymbol{v}_{\max}\right)^2. \tag{14}$$

- Collaboration Penalty R_{collab} : During centralized training, a reward term based on the total number of completed tasks is added, encouraging individual behaviors beneficial to the collective.

The total reward is the weighted sum of each component:

$$r_{i,t} = \omega R_{\text{comp}} + \omega_2 R_{\text{prog}} + \omega_3 R_{\text{rsu}} + \omega_4 R_{\text{speed}} + \omega_5 R_{\text{collab}}, \tag{15}$$

The weights are adjusted through experiments to balance the different objectives.

5) Optimization Problem Formulation

Based on the above MDP model, the goal of each vehicle is to learn an optimal policy π that maximizes its expected discounted cumulative reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\substack{s_0\sim\rho_0 \\ a_t\sim\pi(\cdot|s_t) \\ s_{t+1}\sim P(\cdot|s_t,a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \quad (16)$$

Where $\gamma \in [0, 1)$ is the discount factor, trading off immediate and future rewards.

From a system-level perspective, we expect the joint policy of all vehicles to maximize the following global utility:

Maximizing the system-wide total task completion rate:

$$\mathcal{J}_1 = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}\left(T_i \text{ completed before } \tau_i\right). \quad (17)$$

Minimizing the system average task completion time:

$$\mathcal{J}_2 = \frac{1}{N} \sum_{i=1}^{f} \left(t_{i,comp} - t_{i,gen}\right). \quad (18)$$

Maximizing the average edge resource utilization:

$$\mathcal{J}_3 = \frac{1}{M} \sum_{j=1}^{M} \overline{L_j}. \quad (19)$$

This is a multi-objective optimization problem. This paper, through careful design of the aforementioned single-agent reward function and adoption of a multi-agent training paradigm, enables individuals, while optimizing their own cumulative rewards, to exhibit behaviors that optimize these global objectives. Therefore, the core algorithmic problem transforms into: How to train a deep reinforcement learning model that can effectively handle hybrid action spaces and learn collaborative strategies in a multi-agent environment? The next chapter will detail our proposed SAC-Discrete solution.

IV. IMPROVED SAC-DISCRETE ALGORITHM DESIGN

A. Algorithm Framework Overview

Addressing the hybrid action space joint optimization problem formalized in Chapter 3, this chapter proposes an improved Hybrid Action SAC-Discrete algorithm. The traditional SAC algorithm exhibits excellent performance and stability when handling continuous action spaces, but its native architecture cannot simultaneously output both continuous and discrete actions within a single policy network. Our extended SAC-Discrete redesigns the policy network with a dual-branch architecture to simultaneously output distribution parameters for continuous actions and class probabilities for discrete actions, thereby achieving effective modeling of hybrid action spaces. However, its direct application to VEC joint optimization scenarios still faces challenges including complex state information containing heterogeneous features, the need to finely balance short-term communication gains with long-term travel efficiency, and the requirement to promote collaboration rather than competition in multi-vehicle environments.

To this end, this chapter makes three key improvements to the standard SAC-Discrete algorithm, with the overall algorithm framework shown in Fig. 2.Among them, the hierarchical feature extraction state encoder achieves efficient representation of the 14-dimensional hybrid state information. The dual-branch output hybrid policy network separately generates Gaussian policies for continuous acceleration and categorical policies for discrete offloading decisions, achieving end-to-end differentiable hybrid action sampling through reparameterization techniques. The multi-objective fusion intelligent reward shaping mechanism balances multiple optimization objectives including task completion, resource utilization, and driving efficiency through dynamic weight adjustment. The algorithm adopts the centralized training, decentralized execution paradigm, with each vehicle acting as an independent agent, sharing experience replay buffers and partial network parameters during training to accelerate learning and promote emergence of collaborative strategies.

B. State Encoding and Feature Extraction

The 14-dimensional state vector observed by vehicles contains information with different scales and semantics. Directly inputting it into the network may lead to training instability or slow convergence. Therefore, we designed a hierarchical feature extraction module to preprocess the state as shown in Fig. 3.Among them, the numerical feature normalization layer performs min-max normalization on continuous numerical features such as position, speed, task data size, remaining time, etc., accelerating network convergence and improving numerical stability. The categorical feature embedding layer targets the 3-dimensional "task type" one-hot encoding, mapping it to a low-dimensional dense feature vector through a learnable embedding layer. This enables the network to learn semantic relationships between different task types rather than treating them as independent symbols. The interactive feature concatenation layer concatenates normalized continuous features, embedded categorical features, and environmental features such as channel quality and server load to form a unified feature vector. Subsequently, this vector undergoes nonlinear transformation through two fully connected layers with ReLU activation functions, with the function content as follows:

$$\boldsymbol{h}_1 = \mathrm{ReLU}\left(\boldsymbol{W}_1 \cdot [\boldsymbol{s}_{\text{num}}\,;\boldsymbol{s}_{\text{embed}}\,;\boldsymbol{s}_{\text{env}}\,] + \boldsymbol{b}_1\right), \quad (20)$$

$$\boldsymbol{h}_2 = \mathrm{ReLU}\left(\boldsymbol{W}_2 \cdot \boldsymbol{h}_1 + \boldsymbol{b}_2\right), \quad (21)$$

Where, h_2 serves as a high-level feature representation shared by subsequent policy networks and value networks, capturing key patterns in the state relevant to decision-making.

C. Hybrid Policy Network Design

The policy network $\pi_\theta(A \mid s)$ takes the high-level feature representation h_2 as input and needs to output the joint probability distribution of hybrid action $\left(A^c, A^d\right)$. We designed the dual-branch structure shown in Fig. 4.

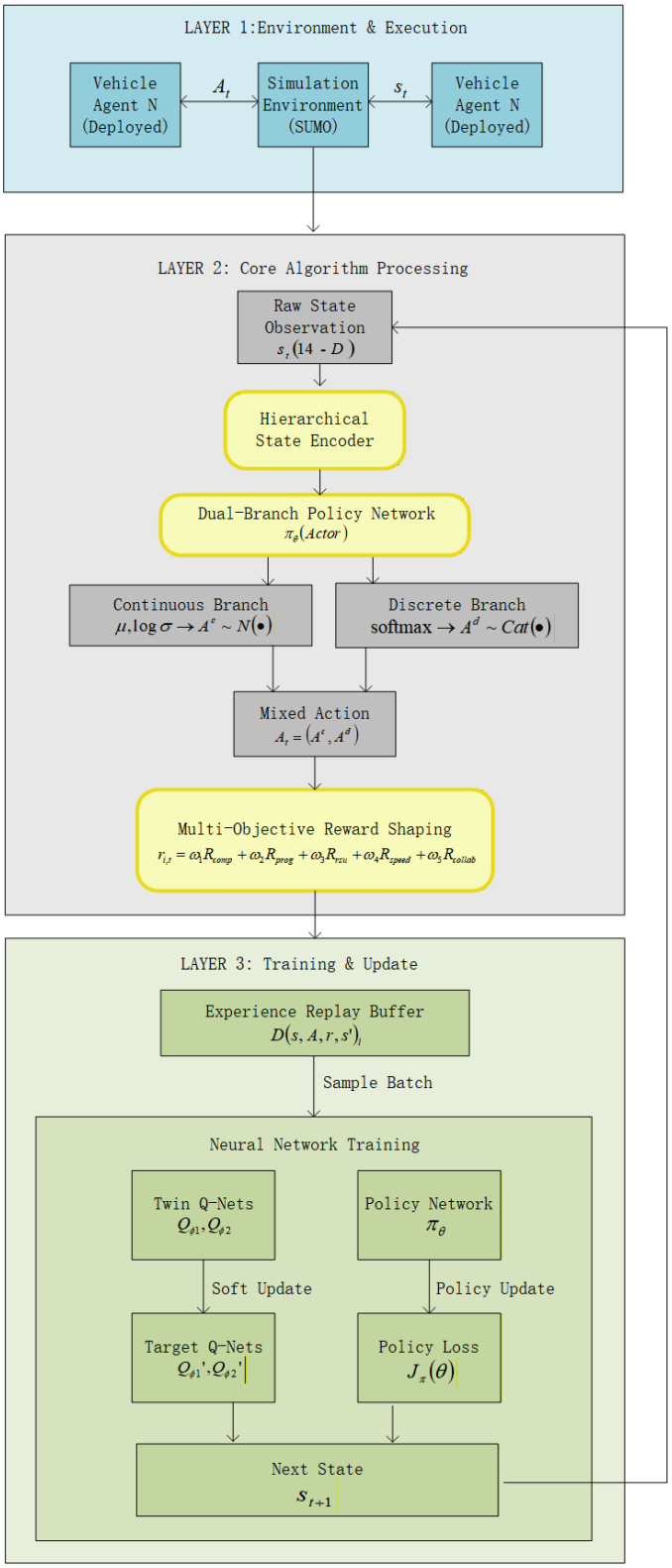

Fig. 2: SAC-D Algorithm Framework.

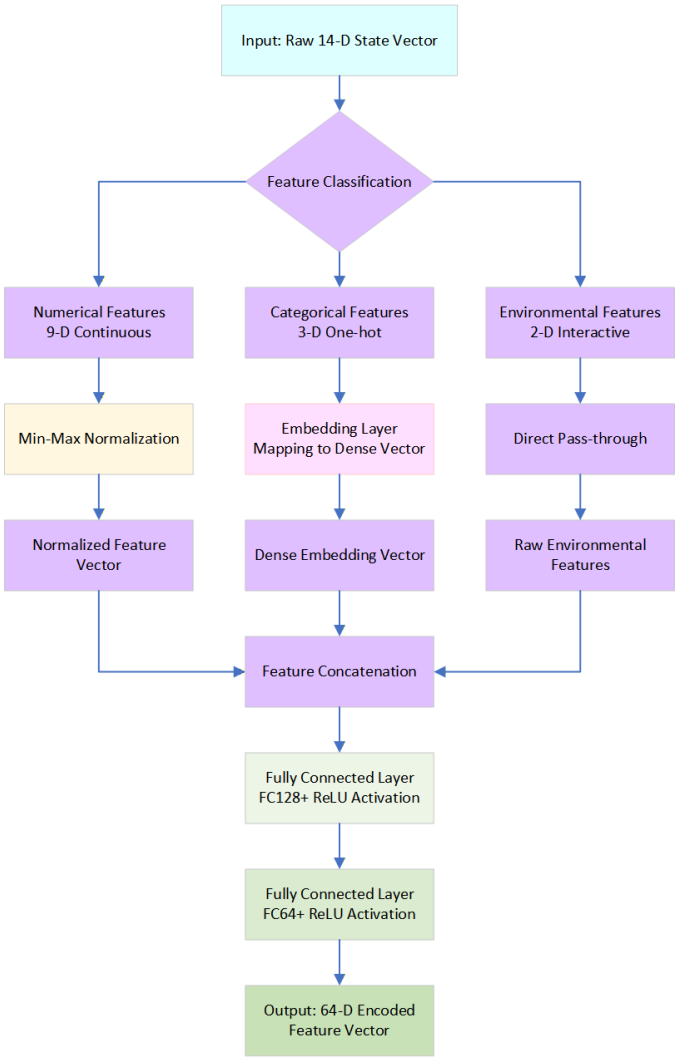

Fig. 3: Hierarchical Feature Extraction Framework.

First is the continuous action branch, responsible for outputting the conditional probability distribution of acceleration A^c . We assume that acceleration follows a Gaussian distribution given the state:

$$A^c \mid s \sim N\left(\mu(s), \sigma^2(s)\right). \tag{22}$$

A fully connected layer is defined as the mean networ $\mu(s)$, taking h_2 as input and outputting a scalar, which is then constrained to the range $[-1, 1]$ through tanh activation, and finally multiplied by the acceleration upper limit a_{max} to obtain the mean:

$$\boldsymbol{\mu} = \boldsymbol{a}_{\max} \cdot \tanh\left(\boldsymbol{W}_\mu \boldsymbol{h}_2 + \boldsymbol{b}_\mu\right). \tag{23}$$

Another fully connected layer is defined as the log standard deviation network $\log\sigma(s)$, also taking h_2 as input and outputting a scalar. To ensure standard deviation is positive and avoid numerical issues, we clip its output:

$$\log\sigma = \text{clamp}\left(W_\sigma h_2 + b_\sigma, \log\sigma_{\min}, \log\sigma_{\max}\right), \tag{24}$$

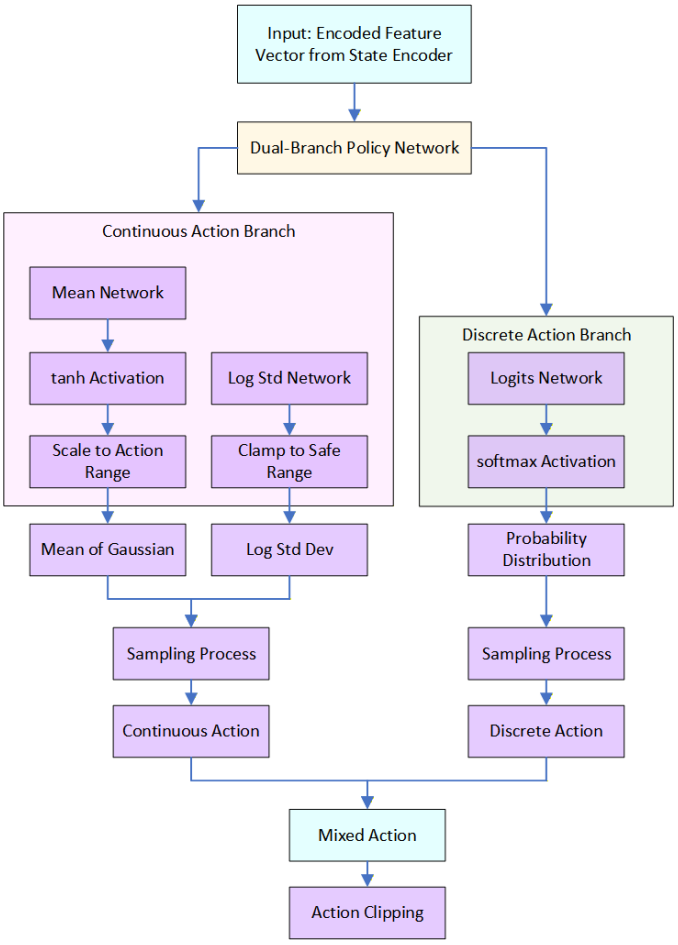

Fig. 4: Dual-Branch Architecture Framework.

Where σ_{min} and σ_{max} are custom small values.

Second is the discrete action branch, responsible for outputting the probability distribution for the three offloading options: local, RSU1, and RSU2. Define the logits network, which is also a fully connected layer, taking h_2 as input but outputting a 3-dimensional vector z :

$$\boldsymbol{z} = \boldsymbol{W}_d \boldsymbol{h}_2 + \boldsymbol{b}_d. \tag{25}$$

The probability distribution of discrete actions is obtained by applying the softmax function to z :

$$\boldsymbol{P}\left(A^d = \boldsymbol{k} \mid \boldsymbol{s}\right) = \frac{\boldsymbol{e}^{z_k}}{\sum_{j=1}^{3} \boldsymbol{e}^{z_j}}. \tag{26}$$

To sample hybrid actions from the policy and compute log probabilities for policy gradient updates, we adopt the following steps:

- Use reparameterization trick for continuous action sampling. First sample noise $\varepsilon \sim N(0,1)$ from a standard Gaussian distribution, then compute $A^c = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\varepsilon}$. Finally, to satisfy practical constraints, clip A^c :

$$A^c = \operatorname{clip}\left(A^c, a_{\min}, a_{\max}\right). \tag{27}$$

- Perform discrete action sampling according to the probability distribution $P\left(A^d \mid s\right)$. During training, to encourage exploration, sampling is typically done directly from this distribution; during evaluation, the action with the highest probability is selected.

The joint log-probability is then obtained by summing the log-probabilities of the continuous and discrete actions:

$$\log \pi(A \mid s) = \log \pi_c\left(A^c \mid s\right) + \log \pi_d\left(A^d \mid s\right), \tag{28}$$

Where, $\log \pi_c\left(A^c \mid s\right)$ is the log probability density function value of the Gaussian distribution, and $\log \pi_d\left(A^d \mid s\right)$ is the log probability value of the categorical distribution. Due to the tanh transformation applied to continuous actions, the log probability requires corresponding Jacobian correction (identical to the approach in standard SAC [22]).

To reduce Q-value overestimation and improve learning stability, we adopt a dual Q-network architecture [23]. Two Q-networks $Q_{\phi 1}$ and $Q_{\phi 2}$ have identical structures but independent parameters. Each Q-network takes the joint representation of state s and hybrid action as input. Since the action is hybrid, we one-hot encode the discrete action, concatenate it with the continuous acceleration value, and then concatenate with state features to form the Q-network input. Specifically, for a given $\left(s, A^c, A^d\right)$, we convert the discrete action A^d into a 3-dimensional one-hot vector o . The input vector is then $[s, A^c, o]$. The Q-network consists of two fully connected layers and an output layer, producing a single scalar Q-value. Target Q-values are computed using Clipped Double Q-learning technique:

$$y = r + \gamma(1-\text{done})\left[\min_{j=1,2} Q_{\phi_j'}\left(s', \tilde{A}'\right) - \alpha \log \pi\left(\tilde{A}' \mid s'\right)\right], \tag{29}$$

Where r is the reward, γ is the discount factor, $done$ is the termination flag, $Q_{\varphi_j'}$ are target Q-networks with parameters ϕ' softly updated from ϕ periodically, $\tilde{A}' = \left(\tilde{A^c}', \tilde{A^a}'\right)$ is the next action sampled from the current policy π, and α is the temperature parameter (entropy coefficient).

The reward function is key to guiding agents toward multi-objective optimization. As described in Chapter 3, our reward function contains multiple components. In algorithm implementation, we concretize it and introduce adaptive mechanisms. To reflect the intelligence of task offloading, we dynamically modify rewards. For example, when a large task is correctly offloaded to an RSU, its base completion reward is multiplied by a factor greater than 1; conversely, if a large task is processed locally and times out, the penalty is increased. This is reflected in the immediate reward r fed back to the agent from the environment. Meanwhile, we adopt automatic entropy adjustment, setting a target entropy value $\aleph$. During training, α is dynamically adjusted by optimizing a loss function with respect to $log\alpha$:

$$L(\alpha) = -\alpha(\log \pi(A \mid s) + \bar{\mathcal{H}}). \tag{30}$$

This ensures policy entropy remains high during early training stages to encourage exploration and gradually decreases later to stabilize policy.

D. Training Process

Algorithm 1 presents the improved SAC-Discrete training process under the Centralized Training Decentralized Execution (CTDE) framework. After training completion, each vehicle only needs to load the trained policy network π_θ during deployment. At each time step, vehicles obtain action distributions through network forward propagation based on their local observation s, then select actions to execute, operating completely distributedly without requiring a central controller or communication with other vehicles. The next chapter will verify the effectiveness of this algorithm in the simulated VEC environment through comprehensive experiments and conduct comparative analysis with multiple baseline methods.

Algorithm 1 Improved SAC-Discrete with CTDE.

Require: Environment $\mathcal{E}$, replay buffer $\mathcal{D}$, networks $\pi_\theta, Q_{\psi_1}, Q_{\psi_2}$
Ensure: Trained policies π_θ

```
Initialize parameters
Initialize target networks Q̄ ← Q
for episode = 1 to K do
    Reset environment, get states {s_i}
    for step = 1 to T do
        for each agent i do
            a_i^acc, a_i^off ← π_θ(s_i)
            Execute, observe s'_i, r_i, d_i
            Store (s_i, a_i, s'_i, r_i, d_i) into D
        end for
        if |D| ≥ B then
            Sample batch from D
            Compute target Q values
            Update critics by minimizing L_Q
            Update actors by minimizing L_π
            Update temperature α if auto_entropy
            Soft update target networks Q̄
        end if
    end for
end for
```

V. Experimental Design and Performance Evaluation

A. Experimental Setup

This chapter aims to systematically verify the performance of the proposed improved SAC-Discrete algorithm on the VEC joint optimization problem. The experimental platform is built based on SUMO 1.24.0 micro-traffic simulator and Python 3.9 environment, with the deep reinforcement learning part implemented using the PyTorch 1.12.0 framework. The simulation strictly follows the scenario defined in Chapter 3, with every vehicle being a task vehicle. Vehicle dynamics are controlled by SUMO's built-in Krauss car-following model. Each vehicle is assigned a random task upon generation, with three task types: small, medium, and large. Computational requirements are positively correlated with data size, and task deadlines increase with task complexity.

The simulation parameters settings for the improved SAC-Discrete algorithm are shown in Table II.

TABLE II: Simulation parameters.

Parameters	Value
Discount factor γ	0.99
Soft update coefficient τ	0.005
Initial temperature α	0.2
Target entropy	-1
Acceleration range	$[-3.0, 3.0]$ m/s^2
Total episodes	60
Max steps per episode	200

To comprehensively evaluate algorithm performance, we use average task completion rate, average travel time, average RSU utilization rate, and cumulative reward curve as quantitative metrics, comparing with the following five baseline methods:

- Greedy Strategy (Greedy): At each decision moment, vehicles select the RSU with the best current channel quality (highest CQ value) and load below threshold 80% for offloading, Otherwise, local processing is performed.
- Offloading Optimization with Fixed Path (OOFP): Vehicles fix a single path, but their offloading decisions are optimized based on a pre-trained SAC-Discrete network. This method isolates path optimization, testing only the effectiveness of offloading decisions.
- Path Optimization with Random Offloading (PORO): This algorithm strips the intelligence of offloading decisions; offloading decisions are completely random, testing only the effectiveness of path and speed optimization.
- Random Strategy (Random): The vehicle acceleration is randomly selected from $[-3, 3]m/s$, the route is chosen arbitrarily, offloading decisions are completely random, representing a non-intelligent baseline.
- SAC: The early version of SAC discretizes the continuous acceleration and treats the entire action space as a single combined discrete action, serving to validate the effectiveness of our proposed improvements.

All baseline methods run under the same simulation environment and task generation settings, with multiple experimental repetitions to obtain statistically reliable results.

B. Experimental Results and Analysis

Fig. 5 shows the cumulative reward learning curve of the SAC-Discrete algorithm over 60 training episodes. It can be observed that the algorithm's reward increases rapidly within approximately the first 15 episodes, indicating that the agent quickly masters basic task completion and driving rules. Between episodes 15 and 40, reward continues to grow slowly amidst fluctuations, suggesting the agent engages in more refined policy exploration and optimization, such as learning when to slightly detour for better channel conditions. After approximately 40 episodes, cumulative reward stabilizes at a high level with small fluctuations, indicating the policy has converged to a stable and high-performance solution.

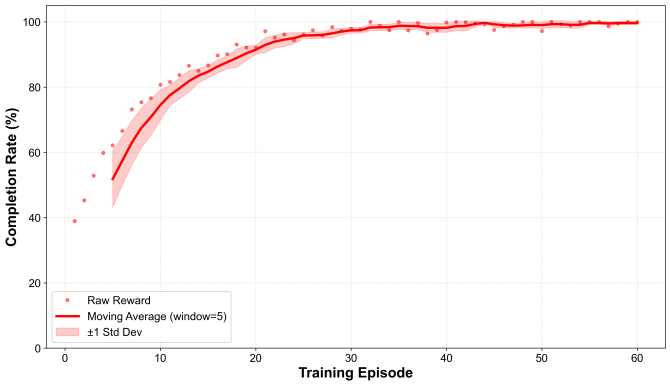

Fig. 5: Convergence of SAC-Discrete.

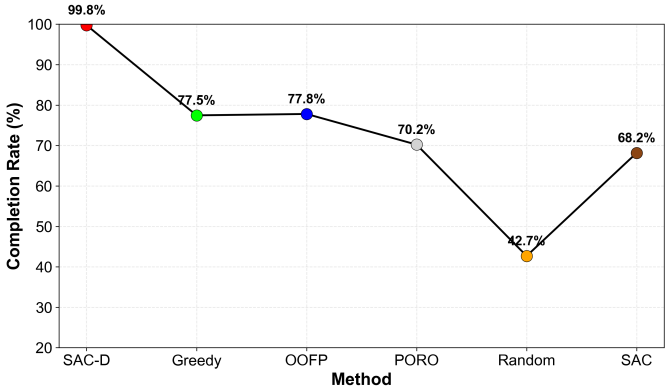

Fig. 6: Task Completion Rate Comparison.

We compared the trained and converged improved SAC-Discrete algorithm with all baseline methods on core metrics.

As shown in Fig. 6 Our method achieves a near-perfect task completion rate of 99.8% , significantly outperforming all baselines. Although the greedy strategy can utilize the current optimal channel, it lacks long-term planning, easily leading to task failure later due to vehicles leaving coverage areas or server overload, achieving only 77.3% completion rate. Fixed path methods suffer from inability to actively approach RSUs, limiting performance. Random methods perform worst.

As shown in Fig. 7 Regarding average travel time, our

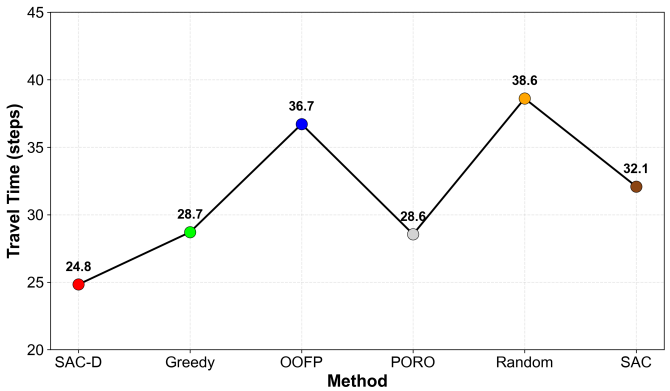

Fig. 7: Average Task Completion Time.

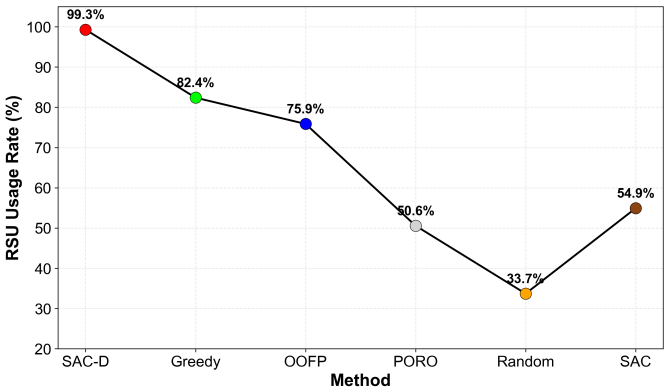

Fig. 8: RSU Utilization Comparison.

algorithm achieves the shortest 25 steps to complete tasks, reducing by 32.2% compared to the fixed path method. This indicates that joint optimization effectively balances the trade-off between "detouring for communication" and "direct fast travel," finding the globally optimal balance point. Furthermore, as shown in Fig. 8 our method achieves 99.3% RSU utilization rate, demonstrating that the algorithm almost always intelligently offloads tasks to appropriate RSUs. In contrast, the 33% low utilization rate of the original method highlights the necessity of the R_rsu term in multi-objective rewards for encouraging edge resource utilization.

To deeply understand the learned policy, we analyzed the agent's decision sequences in typical scenarios. It was found that for large tasks, agents tend to plan paths in advance, proactively drive towards less loaded RSUs, and even start transmission early when channel conditions are acceptable. For small tasks, agents prefer quick local processing while driving, or "conveniently" offload only when passing by RSUs, avoiding detour costs. Throughout the journey, vehicles do not travel at constant speed. When far from RSUs and task processing pressure is low, they appropriately accelerate to shorten travel time; when approaching target RSUs or tasks are about to deadline, they proactively decelerate to extend high-quality communication windows. Simultaneously, agents can avoid overloaded RSUs based on received server load information, directing tasks to idle servers, achieving simple distributed load balancing.

The above experimental results comprehensively confirm the superiority of the improved SAC-Discrete algorithm in solving VEC joint optimization problems. They demonstrate the critical value of placing path planning and offloading decisions within a unified framework for collaborative optimization; the multi-objective reward design successfully guides agents to simultaneously optimize multiple competing objectives; the network design for hybrid states and actions is fundamental to the algorithm's ability to learn complex policies. Its decentralized execution characteristic also aligns with the distributed architecture requirements of V2X networks.

C. Sensitivity Analysis of Reward Weights

The proposed multi-objective reward function 15 plays a crucial role in guiding the agent's learning. The balance

between its components is controlled by the weights ω_1 to ω_5. To evaluate the sensitivity of the policy to these hyperparameters, we conducted a sensitivity analysis focusing on the task completion reward weight ω_1, the driving efficiency reward weight ω_3, and the resource utilization reward weight ω_5. Based on the default configuration of $\omega_1 = 1.0$, $\omega_3 = 0.5$, $\omega_5 = 0.2$, we evaluated the algorithm's performance under four alternative configurations:

- high emphasis on task completion ($\omega_1 = 2.0$).
- low emphasis on task completion ($\omega_1 = 0.5$).
- high emphasis on driving efficiency ($\omega_3 = 1.0$).
- high emphasis on resource utilization ($\omega_5 = 1.0$).

The results reveal a clear trade-off between task completion rate and travel time when varying ω_1. Increasing ω_1 to 2.0 significantly improves the task completion rate from 96.8% to 98.5%, but at the cost of a notable increase in average travel time, as vehicles are more willing to detour or decelerate to secure successful offloading opportunities. Conversely, reducing ω_1 to 0.5 decreases the average travel time by 8.2%, while the task completion rate drops to 88.1%.

When ω_3 is increased to 1.0, the agent prioritizes travel speed, resulting in a moderate reduction in travel time with a slight decrease in task completion rate. For ω_5 increased to 1.0, the policy shows relatively good robustness, maintaining a consistently high completion rate ($> 95\%$) and travel time within a 5% margin of the default configuration. These observations indicate that while the policy is most sensitive to the ω_1-ω_3 trade-off, the algorithm's structure allows for effective learning across a range of hyperparameters, demonstrating its practical flexibility.

VI. DISCUSSION AND FUTURE WORK

A. Discussion on Scalability

While the proposed SAC-Discrete algorithm demonstrates superior performance in the considered two-RSU scenario, its scalability to environments with a significantly larger number of RSUs warrants further discussion. The current dual-branch architecture outputs a discrete action distribution via a softmax layer, whose dimension scales linearly with the number of RSUs N. Theoretically, the algorithm can handle any finite N. However, in practice, a large N leads to a vast discrete action space, which can severely challenge the efficiency of policy exploration and the stability of the Q-value estimation. The performance may degrade as the probability mass is spread across many suboptimal actions.

To address this scalability challenge, future work could explore several promising directions. One approach is to decompose the decision-making process using hierarchical reinforcement learning, where a high-level policy first selects a candidate cluster of RSUs, and a low-level policy then chooses a specific RSU within that cluster. Another avenue is to leverage the parametric nature of the discrete actions. For instance, the action "select RSU j " can be reparameterized by its features, such as location and current load. A policy network could then generate a continuous action that maps to the most suitable RSU based on these features, effectively decoupling the action space size from the network architecture. Furthermore, integrating Graph Neural Networks (GNNs) could allow the agent to process the topology and states of all RSUs in a permutation-invariant manner, enabling the policy to generalize to a varying number of RSUs without architectural changes.

B. Generalization to Complex Scenarios

The current system model simplifies several aspects of the real world while capturing core VEC elements. The road network is a simple one-way road with two junctions. In complex urban environments with multiple lanes, multiple traffic lights, and diverse vehicle densities, the dynamics of vehicles and route choice become more complex. The agent's state representation needs to be enriched using, for example, occupancy grid maps or traffic light phases, which may require more complex encoders such as convolutional neural networks (CNNS). The scalability analysis in the previous section also highlights the challenge of generalizing to a large number of Rsus. Future work will focus on validating and adapting the algorithms in these more realistic and complex urban environments.

VII. CONCLUSION

The convergence of the Internet of Vehicles and edge computing is profoundly reshaping the landscape of future intelligent transportation. Vehicles are no longer isolated mobile nodes but are integrated into a networked intelligent agent system capable of real-time perception, intelligent decision-making, and efficient collaboration. This paper addresses the fundamental and critical problem of "traveling" and "computing" collaboration. By introducing an improved SAC-Discrete algorithm, the significant potential and feasibility of joint optimization are verified in a simulation environment. Research results indicate that through intelligent algorithm design, we can enable vehicles to be not only efficient "travelers" but also intelligent "computers" during their journey to destinations, thereby achieving safer, more efficient, and greener intelligent mobility overall. Although this research has achieved remarkable results, there are still limitations such as simplified scenario models and single road topology. Based on the achievements and limitations of this study, future work can expand in several directions including algorithm extension to more complex scenarios, integration of multi-source information and models, improvement of algorithm practicality and reliability, and exploration of emerging paradigms and applications.

REFERENCES

[1] J. Shi, J. Du, J. Wang, and J. Yuan, "Deep reinforcement learning-based v2v partial computation offloading in vehicular fog computing," in *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, 2021, pp. 1–6.

[2] Q. Wu, H. Liu, R. Wang, P. Fan, Q. Fan, and Z. Li, "Delay-sensitive task offloading in the 802.11p-based vehicular fog computing systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 773–785, 2020.

[3] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[4] W. Saad, M. Bennis, and M. Chen, "A vision of 6g wireless systems: Applications, trends, technologies, and open research problems," *IEEE Network*, vol. 34, no. 3, pp. 134–142, 2020.

[5] K. Zhang, M. Peng, and Y. Sun, "Delay-optimized resource allocation in fog-based vehicular networks," *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1347–1357, 2021.

[6] S. Sharma, S. Ghosh, M. R. Bhatnagar, and B. K. Panigrahi, "Game-theoretic flexible resource allocation for handoff in hybrid v2x communication," in *2023 IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2023, pp. 1–7.

[7] H. Park, Y. Jang, and J.-W. Choi, "Latency analysis of 5g c-v2x real-time video transmission over different channel states," in *2025 IEEE 101st Vehicular Technology Conference (VTC2025-Spring)*, 2025, pp. 1–6.

[8] Y. Eunju, K. Bongseob, S. Geunhan, Y. Kyungsu, and H. Sujin, "V2x traffic-load generator system for enabling efficient vehicle communication load-balancing," in *2023 14th International Conference on Information and Communication Technology Convergence (ICTC)*, 2023, pp. 745–749.

[9] I. Lee and D. K. Kim, "Resource allocation in nr-v2x mode 2 using multi agent dqn," in *2023 Fourteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, 2023, pp. 17–19.

[10] Z. Mlika and S. Cherkaoui, "Deep deterministic policy gradient to minimize the age of information in cellular v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 23 597–23 612, 2022.

[11] S. Yu, S. Li, Y. Li, Y. Fan, Y. Zhou, and J. Peng, "A risk-barrier soft actor-critic approach for automotive intelligent cruise control towards v2x communication," in *2025 9th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2025, pp. 1–6.

[12] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3113–3125, 2019.

[13] Annu and P. Rajalakshmi, "Distance-based queue modeling in sidelink c-v2x communication for platooning: Towards 6g-v2x," in *2024 IEEE 100th Vehicular Technology Conference (VTC2024-Fall)*, 2024, pp. 1–5.

[14] L. Ganeshkumar, R. Ramachandran, and T. Ohtsuki, "Delay-constrained sum-rate maximization in star-ris v2x communications using lyapunov framework," *IEEE Access*, vol. 14, pp. 857–874, 2026.

[15] P. Zhou, X. Xing, X. Wang, T. Wang, and G. Wang, "Adaptive task offloading in high-speed vehicular networks using enhanced d3qn algorithm," in *2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2024, pp. 607–614.

[16] X. Bi, J. Shi, B. Zhang, Z. Lyu, and L. Huang, "An rsu-crossed dependent task offloading scheme for vehicular edge computing based on deep reinforcement learning," *International Journal of Sensor Networks*, vol. 41, no. 4, pp. 244–256, 2023.

[17] K. Moghaddasi and R. Jurdak, "An energy-aware distributed federated soft actor-critic framework for intelligent task offloading in vehicular mobile edge computing networks," *Ad Hoc Networks*, vol. 180, p. 104043, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1570870525002914

[18] V. R. de Lima and M. L. de Campos, "Fully distributed multi-agent processing strategy applied to vehicular networks," *Vehicular Communications*, vol. 49, p. 100806, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209624000810

[19] Q. Zhong, N. Ye, and S. Gao, "A hyper-heuristic algorithm for joint task offloading, resource allocation and trajectory design in uav-assisted vehicular networks," in *International Conference on SmartRail, Traffic and Transportation Engineering*. Springer, 2024, pp. 188–194.

[20] N. Keshari and D. Singh, "Tcv-d: An approach for path selection in vehicular task offloading," *Vehicular Communications*, vol. 47, p. 100770, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209624000457

[21] C. S.-H. Hsu, J. Martín-Pérez, D. De Vleeschauwer, L. Valcarenghi, X. Li, and C. Papagianni, "A deep rl approach on task placement and scaling of edge resources for cellular vehicle-to-network service provisioning," *IEEE Transactions on Network and Service Management*, 2025.

[22] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. Pmlr, 2018, pp. 1861–1870.

[23] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

Jun Wang graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Boda College of Jilin Normal University in 2023. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks and Fog Networks.

Lin Chai graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Lyuliang University in 2022. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks.

Yumei Yang graduated with a Bachelor of Science degree in Mathematics and Applied Mathematics from Yunnan Minzu University in 2024. She is currently pursuing a Master of Science degree in Probability Theory and Mathematical Statistics at Yunnan Minzu University. Her research interests focus on Vehicular Ad-hoc Networks.

Wu Wang received his B.S. and M.S. degrees from Yunnan University, China, in 2003 and 2007, respectively. Received his Ph.D. degree from Future University Hakodate, Japan, in 2018. He is currently an associate professor at the School of Mathematics and Computer Science at Yunnan Minzu University. His research interest includes ad hoc networks, neural network, and network security.