Proceedings of CONF-MSS 2026 Syposium: Mechanical Control and Automation
DOI: 10.54254/2755-2721/2026.BJ32998

CoV2V-MARL: Cooperative Multi-Agent Reinforcement Learning for Task Offloading in V2V Edge Computing Networks

Zelei Zhang

School of Intelligence and Information Engineering, Tangshan University, Tangshan, China
3038627275@qq.com

Abstract. Vehicle-to-vehicle (V2V) communication is becoming an important option for low-latency task processing in vehicular edge computing (VEC). Offloading in this setting, however, is difficult to optimize because network topology changes rapidly, vehicle resources are heterogeneous, and purely self-interested decisions can create congestion and uneven resource use. To address these issues, this paper proposes CoV2V-MARL, a cooperative multi-agent reinforcement learning framework for task offloading in V2V edge computing networks. Each vehicle is treated as an agent, and cooperation is encouraged through a graph-based state representation, a reward function that reflects both local and system-level performance,and a centralized-training decentralized-execution strategy. The graph encoder captures time-varying neighborhood relationships and resource conditions, while the reward design discourages overloaded selections and promotes better load sharing. Experiments in dynamic V2V scenarios show that CoV2V-MARL outperforms several heuristic and reinforcement-learning baselines. Compared with a greedy strategy and single-agent DDPG, the proposed method reduces average task latency by 28.7% and 19.3%, improves energy efficiency by 32.1% and 24.5%, and improves load balancing by 41.2% and 33.7%, respectively. Overall, the results support cooperative learning as a practical approach to scalable task offloading in dynamic V2V environments.

Keywords: Task Offloading, Vehicular Edge Computing, Multi-Agent Reinforcement Learning, Vehicle-to-Vehicle (V2V) Communication

1. Introduction

1.1. Research background

Modern connected vehicles increasingly run applications such as perception, real-time path planning, and augmented reality navigation, all of which require fast and reliable computation [1,2]. Because onboard units (OBUs) have limited processing capacity, a vehicle may not be able to complete every task within its latency budget. Vehicular edge computing (VEC) helps relieve this pressure by allowing tasks to be offloaded to nearby edge resources, such as roadside units (RSUs) or neighboring vehicles, through V2I and V2V links [3]. Among these options, V2V offloading is

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.BJ32998

especially appealing because it can use idle resources in nearby vehicles, reduce reliance on fixed infrastructure, and shorten transmission distance for time-sensitive workloads [4].

In practice, however, V2V collaborative offloading is not easy to manage. Rapid mobility causes frequent topology changes and intermittent links, so a decision that is reasonable at one moment may become ineffective shortly afterward [5]. At the same time, nearby vehicles differ in computing power, residual energy, and link quality, which makes resource allocation more complicated. When vehicles optimize only for themselves, they may repeatedly choose the same favorable nodes, leading to contention, workload imbalance, and poorer overall system performance [6]. For this reason, V2V offloading should be viewed as a coupled sequential decision problem rather than a set of isolated choices. Reinforcement learning (RL) has shown promise for dynamic resource management in edge systems, and multi-agent reinforcement learning (MARL) is a natural fit when many vehicles interact simultaneously [7]. Even so, much of the existing RL literature focuses on V2I settings or assumes limited coupling among agents [8]. Those assumptions make it difficult to represent the kind of cooperation required in dense and highly dynamic V2V networks. This motivates a MARL framework that can explicitly account for changing topology, heterogeneous resources, and coordination among vehicles.

1.2. Related work

1.2.1. V2V collaborative offloading

Early work on V2V task offloading largely relied on heuristics or mathematical optimization. Dai et al. [9] proposed a joint load-balancing and offloading scheme based on current resource conditions, but the decision logic remains largely hand designed. Sahni et al. [10] formulated collaborative offloading as a multi-hop optimization problem and showed the benefit of partial offloading, although solving the optimization becomes costly in dynamic settings. More recent studies have turned to RL for better adaptability. For example, Liu et al. [11] used deep reinforcement learning for vehicular offloading, but their method still approaches the problem mainly from a single-agent perspective and does not explicitly model cooperation among neighboring vehicles.

1.2.2. Multi-agent reinforcement learning in edge computing

MARL has also been explored for distributed resource management in edge computing. Wu et al. [12] combined graph neural networks with MARL to capture structural dependencies among edge nodes, showing that graph-based state representations can be useful in complex environments. Zhan et al. [13] studied decentralized offloading with deep RL and emphasized the importance of agent interaction in larger systems. GAPO [4] further introduced graph attention into vehicular offloading, but its main focus is multi-hop, V2I-assisted scenarios rather than direct V2V collaboration. Taken together, these studies provide a useful foundation, but they still leave room for a cooperative, graph-based MARL design aimed specifically at dynamic V2V computing.

1.3. Research contributions

The main contributions of this study are summarized as follows.

1. We develop CoV2V-MARL, a cooperative multi-agent reinforcement learning framework for V2V collaborative task offloading in which each vehicle acts as a decision-making agent.

2. We introduce a graph-based state encoder and a cooperation-aware reward mechanism to capture neighborhood interactions and align individual offloading actions with system-level

DOI: 10.54254/2755-2721/2026.BJ32998

objectives.

3. We adopt a centralized-training decentralized-execution (CTDE) strategy to improve training stability and generalization in heterogeneous, time-varying V2V environments.

2. System model

2.1. Network architecture

We consider a highway V2V edge computing network composed of a vehicle set $V=\{v_1,v_2,\ldots,v_N\}$ and an RSU set $R=\{r_1,r_2,\ldots,r_M\}$. Each vehicle is equipped with an OBU that has limited computing capacity f_i (in giga-cycles per second) and residual energy E_i (in joules). Vehicles communicate with neighboring vehicles within a communication range d_{max} through the IEEE 802.11p protocol [14]. The available bandwidth between vehicles v_i and v_j at time t is denoted by $b_{i,j}(t)$. RSUs provide supplementary computing support and may also serve as relay nodes in multi-hop communication.

2.2. Task model

Computation tasks are generated dynamically at each vehicle according to a Poisson arrival process with rate λ. A task generated by vehicle v_i is represented as $T_i=(s_i,c_i,\tau_i)$, where s_i is the input data size in KB, c_i is the required number of CPU cycles in giga-cycles, and τ_i is the maximum tolerable completion delay. In this study, tasks are broadly divided into two categories: delay-sensitive tasks, such as perception and control, and computation-intensive tasks, such as data analysis or map processing. This task model reflects the diversity of workload characteristics in practical vehicular applications.

2.3. Communication and computation model

2.3.1. Communication latency

The communication delay between vehicles v_i and v_j consists of transmission delay and queueing delay. The transmission delay for offloading task T_i from v_i to v_j is written as

$$L_{comm}^{i,j}(t)=\frac{s_i}{b_{i,j}(t)} \tag{1}$$

To capture contention on the wireless link, the queueing delay is modeled with an M/M/1 approximation [4]:

$$L_{queue}^{i,j}(t)=\frac{\rho_{i,j}(t)}{b_{i,j}(t)(1-\rho_{i,j}(t)+\varepsilon)} \tag{2}$$

where $\rho_{i,j}(t)$ is the link utilization ratio and ε is a small constant used to avoid numerical instability. The total communication delay is therefore

$$L_{v2v}^{i,j}(t)=L_{comm}^{i,j}(t)+L_{queue}^{i,j}(t) \tag{3}$$

DOI: 10.54254/2755-2721/2026.BJ32998

2.3.2. Computation latency

If task T_i is executed locally at vehicle v_i, the corresponding computation delay is

$$L_{local}^{i} = \frac{c_i}{f_i \alpha_i} \tag{4}$$

where $\alpha_i \in (0,1]$ denotes the fraction of local computing resources assigned to the task. If the task is offloaded to a neighboring vehicle v_j, the remote execution delay becomes

$$L_{offload}^{i,j} = \frac{c_i}{f_j \beta_{i,j}} \tag{5}$$

where $\beta_{i,j} \in (0,1]$ is the fraction of the computing resources of v_j allocated to task T_i.

2.3.3. Total task latency

For full offloading from v_i to v_j, the total delay is the sum of communication delay and remote computation delay:

$$L_{total}^{i,j}(t) = L_{v2v}^{i,j}(t) + L_{offload}^{i,j} \tag{6}$$

If partial offloading is allowed with ratio $\lambda_i \in [0,1]$, the task completion time is determined by the slower branch between local execution and offloaded execution:

$$L_{total}^{i}(t) = max\left(\frac{(1-\lambda_i)c_i}{f_i \alpha_i},\ L_{v2v}^{i,j}(t) + \frac{\lambda_i c_i}{f_j \beta_{i,j}}\right) \tag{7}$$

2.4. Optimization objective

The objective is to jointly optimize service latency, vehicle energy consumption, and load balancing. Let $\bar{L}$ denote the average task completion delay, $\bar{E}$ denote the average energy consumption, and $\bar{B}$ denote the load-balancing index of the network. The overall objective is formulated as

$$minJ = w_1 \overline{L} + w_2 \overline{E} + w_3 (1 - \overline{B}) \tag{8}$$

where w_1, w_2, and w_3 are nonnegative weights satisfying $w_1+w_2+w_3=1$. This formulation allows the proposed framework to balance efficiency, energy cost, and fairness in resource utilization.

3. CoV2V-MARL framework design

3.1. Multi-agent system definition

In the proposed framework, each vehicle is modeled as an agent in a cooperative MARL environment. The system can be described by a tuple $(S,A,P,\mathscr{R},\gamma)$, where S is the state space, A is the action space, P is the transition function, $\mathscr{R}$ is the reward function, and γ is the discount factor.

The state of each agent contains both local and neighborhood information. The local state includes residual energy, current resource utilization, and task queue length. The neighborhood state

DOI: 10.54254/2755-2721/2026.BJ32998

includes the identities of neighboring vehicles, their available computing resources, and the current link bandwidths. Because these variables evolve with vehicle mobility, the state is naturally represented as a dynamic graph in which nodes correspond to vehicles and edges correspond to available V2V links.

The action of an agent includes three components: the offloading target (local execution or a neighboring vehicle), the offloading ratio, and the resource allocation ratio. The reward function combines multiple objectives, including task delay, energy consumption, load balancing, and a cooperation bonus that encourages agents to reduce the burden on overloaded neighbors and improve deadline satisfaction at the system level.

3.2. Graph-based state encoding

To encode the dynamic neighborhood relationships, CoV2V-MARL adopts a graph attention network (GAT) [4]. The encoder aggregates the features of neighboring vehicles with learnable attention weights so that the agent can distinguish more informative neighbors from less relevant ones. A single GAT layer can be expressed as

$$h_i^{(l+1)} = \sigma(\sum_{j \in N_i} \alpha_{ij}^{(l)} W^{(l)} h_j^{(l)}) \tag{9}$$

where $h_i^{(l)}$ is the node embedding of vehicle v_i at layer l, N_i is the neighbor set of v_i, $\alpha_{ij}^{(l)}$ is the attention coefficient between nodes i and j, and $W^{(l)}$ is a learnable transformation matrix. By aggregating both local and neighboring features, the encoder produces a compact representation that captures the dynamic topology and resource status of the V2V network.

3.3. Cooperative policy learning

We adopt a centralized-training decentralized-execution strategy to improve both learning stability and deployment practicality. During training, a centralized critic uses global information from all agents to evaluate the joint system state. During execution, each vehicle makes decisions independently using only its own local observation and encoded neighborhood information. This design improves coordination during learning while preserving the distributed nature of V2V decision making.

The policy is implemented with an actor-critic architecture. The actor network takes the GAT-based state embedding as input and outputs the offloading target, the offloading ratio, and the resource allocation ratio. The centralized critic takes the concatenated embeddings of all agents and estimates the global state value. Advantage values are computed using generalized advantage estimation, and the actor is updated with the PPO objective [15]. This combination allows the framework to maintain stable policy updates under highly dynamic traffic conditions.

3.4. Training algorithm

The training process of CoV2V-MARL consists of five stages. First, the actor and critic networks are initialized together with the replay or trajectory buffer. Second, at each time step, every vehicle observes its current state, encodes the neighborhood graph, selects an offloading action, and interacts with the simulation environment. Third, the collected trajectories are used by the centralized critic to estimate state values and advantages. Fourth, each actor updates its policy with PPO-style clipped optimization, while the critic is updated by minimizing the value estimation error.

DOI: 10.54254/2755-2721/2026.BJ32998

Finally, the interaction and update steps are repeated until the average reward converges. This procedure enables the agents to gradually learn cooperative offloading behavior from repeated system-level feedback.

4. Experimental evaluation

4.1. Experimental setup

4.1.1. Simulation environment

The experiments are conducted in a highway scenario generated by SUMO, and the learning framework is implemented in Python. The road length is 1000 m, and vehicle speed ranges from 60 km/h to 120 km/h. The main simulation settings are listed in Table 1.

Table 1. Parameter settings for different categories

Category	Parameter Setting
Vehicles	40–80 vehicles; OBU capacity 0.25–0.5 Gcps; initial energy 1000 J
RSUs	5 RSUs; coverage radius 200 m; computing capacity 20–35 Gcps
Tasks	Data size 512–1024 KB; complexity 0.5–1.5 Gc; arrival rate 0.1–0.02 s^-1
Communication	Bandwidth 10–30 Mbps; V2V communication range 200 m

4.1.2. Baseline algorithms

To assess the effectiveness of the proposed method, we compare CoV2V-MARL with several representative baselines, as listed in Table 2.

Table 2. Different baseline model and descriptions

Baseline	Description
Greedy Lowest Load	Chooses the neighboring vehicle with the lowest current CPU utilization
Single-Agent DDPG	Each vehicle makes offloading decisions independently without cooperation
MARL without Cooperation	Multi-agent learning with individual rewards only and no cooperation bonus
GAPO [4]	Graph attention-based RL designed mainly for V2I-oriented multi-hop offloading

4.2. Performance metrics

Four metrics are used in the experiments. Average task completion latency measures the time from task generation to task completion. Average energy consumption captures the cost of communication and computation. Load-balancing degree reflects how evenly workload is distributed across vehicles. Task completion rate measures the fraction of tasks finished within their delay deadlines. Together, these metrics reflect timeliness, efficiency, and overall service quality.

DOI: 10.54254/2755-2721/2026.BJ32998

4.3. Experimental results

4.3.1. Performance under different vehicle numbers

As the number of vehicles grows from 40 to 80, performance degrades for all methods because contention becomes more severe. Even under this heavier load, CoV2V-MARL remains the best-performing method. It lowers average latency by 28.7% relative to the greedy baseline, by 19.3% relative to single-agent DDPG, and by 12.6% relative to GAPO. The load-balancing index also improves by 41.2% over the greedy method and by 33.7% over single-agent DDPG. The gap becomes more noticeable as the network gets denser, which suggests that cooperative learning is particularly helpful when vehicles compete for limited resources.

4.3.2. Performance under different task arrival rates

Across the tested task-arrival settings, the system experiences varying computational pressure, and the advantage of cooperative decision making becomes clearer under heavier load. In these cases, CoV2V-MARL reduces average energy consumption by 32.1% compared with the greedy method and by 24.5% compared with single-agent DDPG. Its task completion rate reaches 92.3%, exceeding the greedy baseline by 18.5% and single-agent DDPG by 12.7%. This pattern indicates that the cooperation-aware policy uses available resources more efficiently and is better at avoiding deadline violations when workload increases.

4.3.3. Ablation study

To isolate the contribution of each design component, we conduct ablation experiments by removing one key module at a time. The main observations are summarized below.

- When the GAT encoder is removed, average latency increases by 15.4%, suggesting that topology-aware state encoding helps the agent identify suitable offloading neighbors.
- When the cooperation reward is removed, load-balancing performance drops by 27.8%, indicating that explicit cooperative incentives are important for aligning local actions with system-level efficiency.
- Without CTDE training, the learning process becomes noticeably less stable and latency fluctuates by about ±12.3%, highlighting the value of coordinated training in a non-stationary multi-agent environment.

Taken together, the ablation results show that the three main design choices are complementary. The graph encoder improves situational awareness, the cooperation reward aligns local actions with the global objective, and CTDE stabilizes the learning process.

4.4. Discussion

The results point to three complementary reasons for the gain of CoV2V-MARL. First, the graph encoder helps each vehicle identify useful neighbors in a rapidly changing topology. Second, the cooperation-aware reward discourages multiple agents from repeatedly selecting the same favorable nodes, which reduces contention and improves fairness. Third, CTDE improves training stability in a decentralized multi-agent setting. That said, the current evaluation is limited to a controlled highway scenario. Testing the framework at urban intersections, under mixed traffic densities, and with less reliable communication would make the empirical study stronger.

DOI: 10.54254/2755-2721/2026.BJ32998

5. Conclusion

In this paper, we proposed CoV2V-MARL, a cooperative multi-agent reinforcement learning framework for task offloading in V2V edge computing networks. The framework combines graph-based state encoding, a cooperation-aware reward, and CTDE training to handle changing topology and heterogeneous vehicle resources. Simulation results show consistent gains in latency, energy efficiency, task completion rate, and load balancing over both heuristic and reinforcement-learning baselines. Future work will extend the framework to more complex urban scenarios, incorporate security and trust considerations, and evaluate performance under more realistic communication uncertainty.

References

[1] Mach P, Becvar Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading [J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.
[2] Mao Y, You C, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective [J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.
[3] Yousefpour A, Fung C, Nguyen T, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey [J]. Journal of Systems Architecture, 2019, 98: 289-330.
[4] Zhao H, Li X, Li C, et al. GAPO: A Graph Attention-Based Reinforcement Learning Algorithm for Congestion-Aware Task Offloading in Multi-Hop Vehicular Edge Computing [J]. Sensors, 2025, 25(15): 4838.
[5] Ke H, Wang J, Deng L, et al. Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks [J]. IEEE Transactions on Vehicular Technology, 2020, 69(7): 7916-7929.
[6] Liang Y, He Y, Zhong X. Decentralized computation offloading and resource allocation in MEC by deep reinforcement learning [C]//2020 IEEE/CIC International Conference on Communications in China (ICCC). IEEE, 2020: 244-249.
[7] Luong N C, Hoang D T, Gong S, et al. Applications of deep reinforcement learning in communications and networking: A survey [J]. IEEE Communications Surveys & Tutorials, 2019, 21(4): 3133-3174.
[8] Chen S, Li W, Sun J, et al. An Efficient Collaborative Task Offloading Approach Based on Multi-Objective Algorithm in MEC-Assisted Vehicular Networks [J]. IEEE Transactions on Vehicular Technology, 2025, 74(10): 11249-11263.
[9] Dai Y, Xu D, Maharjan S, et al. Joint Load Balancing and Offloading in Vehicular Edge Computing and Networks [J]. IEEE Internet of Things Journal, 2018, 6(2): 4377-4387.
[10] Sahni Y, Cao J, Yang L, et al. Multi-Hop Multi-Task Partial Computation Offloading in Collaborative Edge Computing [J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(5): 1133-1145.
[11] Liu X, Gao C, Zhao J, et al. Deep Reinforcement Learning for Task Offloading in Vehicular Edge Computing [C]//2024 IEEE International Conference on Communications (ICC). IEEE, 2024: 1-6.
[12] Wu J, Zou Y, Zhang X, et al. Dependency-Aware Task Offloading Strategy via Heterogeneous Graph Neural Network and Deep Reinforcement Learning [J]. IEEE Internet of Things Journal, 2025, 12(24): 22915-22933.
[13] Zhan Y, Guo S, Li P, et al. A deep reinforcement learning based offloading game in edge computing [J]. IEEE Transactions on Computers, 2020, 69(6): 883-893.
[14] Zhao H, Li X, Li C, et al. GAPO: A Graph Attention-Based Reinforcement Learning Algorithm for Congestion-Aware Task Offloading in Multi-Hop Vehicular Edge Computing [J]. Sensors, 2025, 25(15): 4838.
[15] Lowe R, Wu Y, Tamar A, et al. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments [C]//Advances in Neural Information Processing Systems. 2017: 6379-6390.