Multi-agent Reinforcement Learning Based Collaborative Multi-task Scheduling for Vehicular Edge Computing

Peisong Li[1], Ziren Xiao[1], Xinheng Wang[1](✉), Kaizhu Huang[2], Yi Huang[3], and Andrei Tchernykh[4]

[1] School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou 215123, China
{peisong.li20,ziren.xiao20}@student.xjtlu.edu.cn, xinheng.wang@xjtlu.edu.cn

[2] Data Science Research Center and Division of Natural and Applied Sciences, Duke Kunshan University, Suzhou 215316, China
kaizhu.huang@dukekunshan.edu.cn

[3] Department of Electrical Engineering and Electronics, University of Liverpool, Liverpool L69 3BX, UK
yi.huang@liverpool.ac.uk

[4] CICESE Research Center, carr. Tijuana-Ensenada 3918, 22860 Ensenada, BC, Mexico
chernykh@cicese.mx

Abstract. Nowadays, connected vehicles equipped with advanced computing and communication capabilities are increasingly viewed as mobile computing platforms capable of offering various in-vehicle services, including but not limited to autonomous driving, collision avoidance, and parking assistance. However, providing these time-sensitive services requires the fusion of multi-task processing results from multiple sensors in connected vehicles, which poses a significant challenge to designing an effective task scheduling strategy that can minimize service requests' completion time and reduce vehicles' energy consumption. In this paper, a multi-agent reinforcement learning-based collaborative multi-task scheduling method is proposed to achieve a joint optimization on completion time and energy consumption. Firstly, the reinforcement learning-based scheduling method can allocate multiple tasks dynamically according to the dynamic-changing environment. Then, a cloud-edge-end collaboration scheme is designed to complete the tasks efficiently. Furthermore, the transmission power can be adjusted based on the position and mobility of vehicles to reduce energy consumption. The experimental results demonstrate that the designed task scheduling method outperforms benchmark methods in terms of comprehensive performance.

Keywords: Multi-agent reinforcement learning · Vehicular edge computing · Multi-task scheduling · Cloud-edge-end collaboration

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 563, pp. 3–22, 2024.
https://doi.org/10.1007/978-3-031-54531-3_1

1 Introduction

Advancements in computing technologies and the widespread implementation of communication infrastructure have led to significant progress in the field of connected vehicles [1]. Connected vehicles are equipped with technology that enables communication with other vehicles, base stations, and the Internet, making them a rapidly evolving area. In addition to their traditional transportation role, connected vehicles are emerging as a mobile computing platform that offers a broad range of services to enhance safety, efficiency, and convenience for drivers and passengers. These services include navigation, entertainment, safety services, and so on [9].

The integration of task processing results from various onboard sensors is crucial for many in-vehicle services. Multi-task fusion, which involves combining data from cameras, Light Detection and Ranging (LiDAR), radar, ultrasonic sensors, and other sources, can provide a comprehensive understanding of the environment around the vehicle [7]. For instance, Advanced Driver Assistance Systems (ADAS) can utilize these sensors to enhance driver safety and offer additional assistance in driving, including adaptive cruise control, lane departure warning, and blind spot monitoring.

For these multiple sensors-supported services, completion time and energy consumption are both critical factors for guaranteeing the Quality of Service (QoS). Multi-task fusion requires processing large amounts of data from multiple sensors in real-time, and any delays in this process could lead to accidents or other safety issues. Additionally, energy consumption is an important consideration for autonomous vehicles, as they rely on a variety of sensors and computing systems that require significant amounts of energy. In this context, task scheduling is critical for connected vehicles. It involves allocating computing resources and offloading tasks. Effective task scheduling can help to improve the performance of connected vehicles by optimizing resource allocation and reducing latency. By prioritizing critical tasks and allocating computing resources based on their processing requirements, task scheduling can ensure that the services are delivered with minimal delay and maximum efficiency. However, task scheduling and resource allocation can be challenging due to the dynamic nature of vehicular environments.

In order to achieve an efficient task scheduling, vehicular edge computing (VEC) has been proposed and implemented [10]. VEC utilizes edge computing technology in connected vehicles to perform processing and storage closer to the data source, reducing latency and improving real-time response. Many task scheduling methods have been proposed for VEC in existing studies: In [16], a Genetic Algorithm-based collaborative task offloading model was formulated to offload the tasks to the base station. In [2], a Particle Swarm Optimization-based task scheduling method was proposed to address the problem of task offloading due to vehicle movements and limited edge coverage. These heuristic algorithms are rule-based and can be used to allocate tasks based on predefined rules, but they may not be able to handle complex and dynamic allocation problems that require more flexible and adaptive solutions. Recently, reinforcement learning was used to optimize the allocation of tasks by modeling the allocation problem as a Markov decision process (MDP) and training the agent to make optimal task scheduling decisions [6]. However, little research has focused on the simultaneous scheduling of multiple tasks generated from one vehicle and joint optimization of completion time and energy consumption.

In order to address the aforementioned issues, a collaborative multi-task scheduling method is proposed that employs a Multi-agent Reinforcement Learning method and cloud-edge-end collaboration architecture. Multi-agent reinforcement learning algorithms can learn from experience and adapt to changes in the environment or task requirements, providing a more flexible and adaptive solution. Among all the existing DRL algorithms, Proximal Policy Optimization (PPO) [14] is a new innovation known for being stable and scalable. Multi-Agent Proximal Policy Optimization (MAPPO) is a variant of the PPO algorithm designed for multi-agent reinforcement learning scenarios. In addition, cloud-edge-end collaboration provides a distributed computing architecture that leverages the strengths of each tier of computing resources to provide efficient task offloading and data processing in changing environmental conditions.

Specifically, the proposed multi-agent PPO algorithm-based task scheduling method is utilized to dynamically assign multiple computation tasks generated from one vehicle to other computing entities in real time. These entities include edge servers, idle vehicles, and the cloud server, and the allocation takes into account the tasks' characteristics, the current environmental state, and the workload of available computing entities. Additionally, the transmission power is adjusted according to the distances between vehicles and other computing entities to reduce the energy consumption for vehicles.

The main contributions are summarised as follows:

(1) A scheme for joint optimization with multiple objectives has been proposed, aiming to minimize both the completion time of service requests and the energy consumption of vehicles.
(2) A multi-agent PPO-based task scheduling and resource allocation method is proposed, which can dynamically allocate multiple tasks based on the changing environment and adjust transmission power according to the distance between the current vehicle and the computing entity. In addition, it allows vehicles to act independently and simultaneously in a decentralized manner, which reduces the need for centralized coordination and communication. This improves scalability and reduces the complexity of multi-agent systems.
(3) A cloud-edge-end collaboration scheme is designed, in which the vehicles can offload tasks to other vehicles, edge servers, and the cloud server in a collaborative way.

2 Related Work

In this section, the existing studies related to collaborative task scheduling and resource allocation in VEC are discussed.

In [3], a joint task offloading and resource allocation scheme for Multi-access Edge Computing scenarios was proposed. This scheme employs parked and moving vehicles as resources to enhance task processing performance and reduce the workload of edge servers. In [5], a joint secure offloading and resource allocation scheme was proposed for VEC networks, utilizing physical layer security and spectrum sharing architecture to improve secrecy performance and resource efficiency, with the aim of minimizing system processing delay. These studies commonly focused on transmission and computation resource allocation but little research focuses on the adjustment of transmission power. The power consumption of the wireless communication system is directly influenced by the transmission power, potentially affecting the battery life of connected vehicles [11].

In [18], a non-orthogonal multiple access based architecture for VEC was proposed, where various edge nodes cooperate to process tasks in real-time. In [4], a collaboration scheme between mobile edge computing and cloud computing is presented to process tasks in Internet of Vehicles (IoV), and a deep reinforcement learning technique is introduced to jointly optimize computation offloading and resource allocation for minimizing the system cost of processing tasks subject to constraints. These studies can effectively offload tasks to the edge server in a collaborative way. However, little research focuses on cooperative task processing among vehicles based on vehicle-to-vehicle (V2V) communications.

In [17], a mobile edge computing-enabled vehicular network with aerial-terrestrial connectivity for computation offloading and network access was designed, with the objective of minimizing the total computation and communication overhead, solved through a decentralized value-iteration based reinforcement learning approach. In [8], a collaborative computing framework for VEC was proposed to optimize task and resource scheduling for distributed resources in vehicles, edge servers, and the cloud, using an asynchronous deep reinforcement algorithm to maximize the system utility. These studies have focused on task scheduling in a dynamic environment but did not consider the simultaneous offloading of multiple tasks from one vehicle, which is a more complex issue that requires optimization in the current VEC scenario.

In order to address the above-mentioned problem, a multi-agent reinforcement learning-based multi-task simultaneous scheduling method is proposed in this paper. With the dynamic adjustment of transmission power and collaboration of computing entities, vehicular energy consumption and request completion time can be optimized jointly.

3 System Model and Problem Definition

In this section, the system model of VEC scenario is illustrated and then the optimization target is formulated.

3.1 Network Model

Figure 1 illustrates the VEC system. The three-tier VEC system is a hierarchical architecture comprising vehicles, edge servers, and a cloud server, each playing a distinct role in data processing and service delivery. At the bottom tier, vehicles act as the local computing nodes, equipped with onboard sensors and processing capabilities for immediate data analysis. The middle tier consists of edge servers strategically positioned within the vehicular network, serving as intermediate processing hubs. The top tier involves a central cloud server that acts as a centralized resource pool, offering extensive computational power and storage capacity. The cloud server supports complex and resource-intensive applications, allowing for scalable and efficient computing solutions across the entire vehicular network. This three-tier structure optimally balances computing resources and response times, ensuring a seamless and responsive VEC environment.

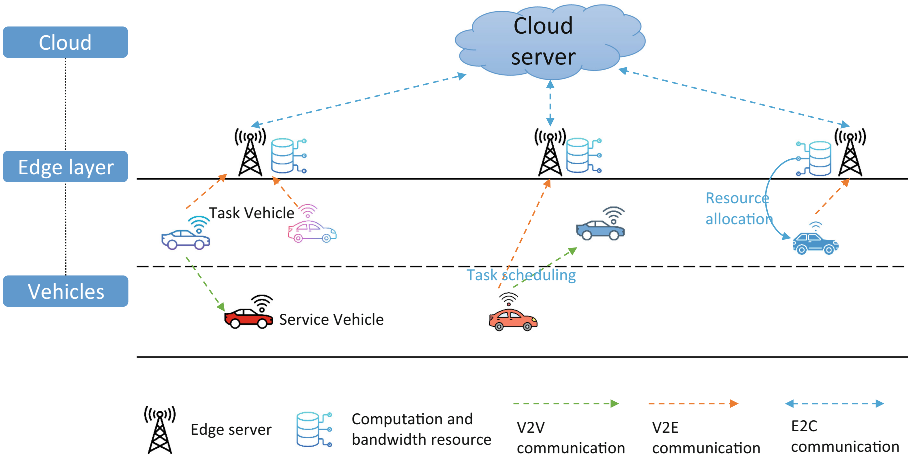


Fig. 1. System architecture

3.2 Mobility Model

Due to the perpetual movement and varying speeds, the positions of vehicles and the distances between them are in constant flux. This dynamic scenario affects the connectivity among vehicles and the choice of the computing entities to which tasks are offloaded [12]. Consequently, mobility emerges as a crucial factor that necessitates consideration in this investigation. In this study, it is assumed that vehicles are in motion on a highway, a standard two-way road.

For each vehicle, coordinate $(x(\tau), y(\tau))$ is used to denote the location at time τ on a 2D map. Thus, the distance between two vehicles i and j can be calculated by (1):

$$d_{i,j}(\tau) = \sqrt{(x_i(\tau) - x_j(\tau))^2 + ((y_i(\tau) - y_j(\tau))^2} \tag{1}$$

Similarly, the distance $d_{i,r}(\tau)$ between the vehicle i and the edge server r can be calculated by (2):

$$d_{i,r}(\tau) = \sqrt{(x_i(\tau) - x_r(\tau))^2 + ((y_i(\tau) - y_r(\tau))^2} \tag{2}$$

3.3 Communication Model

In this study, we dynamically adjust the transmission power over time, taking into account the distances between computing entities. This adjustment directly influences the power consumption of the wireless communication system, which, in turn, can impact the battery life of the connected vehicles. Increasing the transmission power extends the communication range, enabling vehicles to communicate over greater distances. However, this extension comes at a cost - higher power consumption, leading to faster battery drain and a reduction in the vehicle's driving range and overall performance. Consequently, determining an optimal transmission power becomes crucial to ensure reliable and efficient communication among connected vehicles.

V2V Data Transmission Rate. The data transmission rate, often measured in bits per second (bps) or a similar unit, is influenced by the quality of the communication channel. A higher Signal-to-Noise Ratio (SNR) allows for a higher data transmission rate because the system can reliably transmit more bits without errors. Conversely, a lower SNR may limit the achievable data transmission rate due to the increased likelihood of errors and the need for error-correction mechanisms, which can reduce the effective data rate.

Firstly, the SNR between the vehicles i and j can be calculated by (3):

$$SNR_{i,j}(\tau) = \frac{p_{i,j}(\tau) \cdot G_{i,j}(\tau)}{\xi_{i,j}(\tau) \cdot d_{i,j}(\tau) \cdot \sigma^2} \tag{3}$$

where $p_{i,j}(\tau)$ denotes the transmission power when transmitting tasks from vehicle i to vehicle j, $G_{i,j}(\tau)$ denotes the channel gain, $\xi_{i,j}(\tau)$ denotes the path loss, σ^2 denotes the white Gaussian noise, and $d_{i,j}(\tau)$ denotes the distance between two vehicles.

Therefore, the data transmission rate $R_{i,j}$ between vehicles i and j can be calculated by (4):

$$R_{i,j}(\tau) = b_{i,j}(\tau) \log_2 \left(1 + SNR_{i,j}(\tau)\right) \tag{4}$$

where $b_{i,j}(\tau)$ represents the allocated bandwidth resource between vehicle i and vehicle j.

V2E Data Transmission Rate. The V2E communication can be significantly influenced by interference. Interference refers to unwanted signals or noise that disrupt the intended transmission between communication devices. It can lead to degraded signal quality, increased error rates, and reduced data transmission rates. In wireless communication systems, interference from other devices, environmental factors, or overlapping signals can adversely affect the reliability and performance of the communication link. Managing and mitigating interference is crucial to maintaining clear and effective communication in vehicular networks. Thus, the interference noise $I_{i,r}(\tau)$ can be calculated by (5):

$$I_{i,r}(\tau) = (N_r(\tau) - 1) \cdot p_{i,r}(\tau) \cdot g_r \tag{5}$$

where $(N_r(\tau) - 1)$ denotes the number of vehicles connecting to the edge server r (except the vehicle i itself), $p_{i,r}(\tau)$ denotes the allocated transmission power to the vehicle i, and g_r denotes the channel gain.

Then, according to the calculated interference noise, the V2E data transmission rate can be calculated by (6):

$$R_{i,r}(\tau) = \eta_i \cdot B_r \cdot \log_2{(1 + \frac{p_{i,r}(\tau) g_r}{I_{i,r}(\tau) + \sigma^2})} \tag{6}$$

where B_r denotes the total available bandwidth of edge server r, η_i denotes the proportion of the bandwidth allocated to vehicle i.

E2C Data Transmission Rate. Similarly, the interference noise $I_{r,cs}(\tau)$ can be calculated by (7):

$$I_{r,cs}(\tau) = \sum_{s \neq r, s \in S} p_s(\tau) \cdot g_r(\tau) \tag{7}$$

Then, according to the calculated interference noise, the E2C data transmission rate can be calculated by (8):

$$R_{r,cs}(\tau) = B_{r,cs} \cdot \log_2{(1 + \frac{p_r(\tau) g_r(\tau)}{I_{r,cs}(\tau) + \sigma^2})} \tag{8}$$

3.4 Computation Model

Task Scheduling. Firstly, the computation tasks are generated by various sensors mounted on the vehicle. Then, the tasks are combined into a task processing request and forwarded to the *Runtime Optimizer*, which can generate task scheduling decisions based on the proposed scheduling method. Following that, the tasks are assigned to multiple computing entities according to the scheduling decision. Finally, the computation results are fused together and then fed back.

In this work, the number of vehicles and edge servers can be denoted by the set $\{1, 2, \cdots, v\}$ and set $\{1, 2, \cdots, s\}$, respectively. Each vehicle generates service requests continuously, with K computation tasks in each request. For each task, the scheduling direction can be denoted by α, where $\alpha \in \{0, 1, 2, 3\}$, $\alpha = 0$ denotes that the task is offloaded to the server, $\alpha = 1$ denotes that the task is allocated to another vehicle, $\alpha = 2$ denotes that the task is executed locally, $\alpha = 3$ denotes that the task is processed in the cloud. In addition, the property of each task can be donated by a tuple $\langle C, S, D\rangle$, where C represents the required CPU cycles to complete the task, S denotes the data size, and D represents the maximal processing time of the computing task.

After task scheduling, the tasks in the vehicle v at the moment can be represented by the set $L_v = \{l_1, l_2, \ldots, l_{ns}\}$, L_v involves the local tasks generated by the vehicle v itself and the scheduled tasks offloaded from other vehicles. Similarly, the tasks offloaded to the edge server s can be denoted by set $G_s = \{g_1, g_2, \ldots, g_{ms}\}$.

Local Computing. For local computing, The task completion time $T^L_{v,k}$ can be calculated according to (9):

$$T^L_{v,k} = T_{v,l_{ns}} + \frac{C_{v,k}}{f_v} \tag{9}$$

The energy consumption $E^L_{v,k}$ can be calculated according to (10):

$$E^L_{v,k} = \xi \cdot (f_v)^\gamma \cdot C_{v,k} \tag{10}$$

where $T_{v,l_{ns}}$ denotes the expected completion time of the tasks in the processing queue, l_{ns} denotes the ns^{th} task at vehicle v, f_v represents the computing power of the vehicle v, which is defined as the number of CPU cycles executed every second. $\xi = 10^{-11}$ and $\gamma = 2$ represent the power consumption coefficients, which are constants.

Edge Computing. Edge computing serves as an extension to local computing, allowing vehicles to offload and process data at edge servers, enhancing computational capabilities and enabling real-time, resource-efficient decision-making. For the offloaded task k, the execution time $T^O_{v,k}$ is comprised of task transmission time $T^{O,trans}_{v,k}$ and task execution time $T^{O,exe}_{v,k}$:

$$T^O_{v,k} = T^{O,trans}_{v,k} + T^{O,exe}_{v,k} \tag{11}$$

In (11), the task transmission time $T^{O,trans}_{v,k}$ can be calculated by:

$$T^{O,trans}_{v,k} = \begin{cases} T^{V2V,trans}_{j,l_{ns}} + \frac{S_{v,k}}{R_{v,j}}, & \alpha_{v,k} = 1 \\ T^{V2E,trans}_{r,g_{ms}} + \frac{S_{v,k}}{R_{v,s}}, & \alpha_{v,k} = 0 \end{cases} \tag{12}$$

where $T^{V2V,trans}_{j,l_{ns}}$ and $T^{V2E,trans}_{s,g_{ms}}$ represent the transmission completion time of the preceding task at vehicle j and edge server s, respectively. l_{ns} and g_{ms} represent the index of the preceding task at vehicle j and edge server s, respectively.

The task execution time $T^{V2E,exe}_{v,k}$ is defined by:

$$T^{O,exe}_{v,k} = \begin{cases} T^{V2V,exe}_{j,l_{ns}} + \frac{C_{v,k}}{f_j}, & \alpha_{v,k} = 1 \\ T^{V2E,exe}_{s,g_{ms}} + \frac{C_{v,k}}{f_s}, & \alpha_{v,k} = 0 \end{cases} \tag{13}$$

where $T^{V2V,exe}_{j,l_{ns}}$ and $T^{V2E,exe}_{s,g_{ms}}$ represent the completion time of the preceding task at vehicle j and edge server s, respectively, l_{ns} and g_{ms} represent the index of the preceding task.

In addition to the task completion time, the energy consumption of the task computation can be calculated by:

$$E^{O}_{v,k} = \begin{cases} p_v \cdot \frac{S_{v,k}}{R_{v,j}} + \xi \cdot (f_j)^{\gamma} \cdot C_{v,k}, & \alpha_{v,k} = 1 \\ p_v \cdot \frac{S_{v,k}}{R_{v,s}}, & \alpha_{v,k} = 0 \end{cases} \tag{14}$$

where p_v is the transmission power of vehicle v, representing the amount of data transmitted per second.

Cloud Computing. Cloud computing functions as an extension to edge computing, providing additional computational resources and storage capabilities for time-tolerant and computation-intensive tasks, enabling scalable processing and storage solutions for vehicular applications. This hierarchical architecture allows for a seamless integration of local, edge, and cloud resources to meet the diverse computing needs in vehicular environments. The total execution time of task k is expressed by:

$$T^{CS}_{v,k} = T^{V2E,trans}_{v,k} + T^{E2C,trans}_{v,k} + T^{CS,exe}_{v,k} \tag{15}$$

where $T^{V2E,trans}_{v,k}$ and $T^{E2C,trans}_{v,k}$ represent the task transmission time from vehicle v to the edge server and from the edge server to the cloud server, respectively, $T^{CS,exe}_{v,k}$ represents the task processing time on the cloud server.

In (15), the task transmission time $T^{E2C,trans}_{v,k}$ can be calculated by:

$$T^{E2C,trans}_{v,k} = T^{E2C,trans}_{cs,h_{ls}} + \frac{S_{v,k}}{R_{r,cs}} \tag{16}$$

where $T^{E2C,trans}_{cs,h_{ls}}$ represents the transmission completion time of the preceding task at cloud server cs.

The task execution time can be calculated by:

$$T^{E2C,exe}_{v,k} = T^{E2C,exe}_{cs,h_{ls}} + \frac{C_{v,k}}{f_s} \tag{17}$$

where $T^{E2C,exe}_{cs,h_{ls}}$ represents the completion time of the preceding task at the cloud server cs.

In the designed system model, the following assumptions were made to simplify the model: (1) The moving of vehicles is disregarded during the task offloading and results feedback. (2) The feedback time is neglected due to the small size of computed results [17].

3.5 Problem Formulation

The objective function serves as a quantifiable measure of performance for the proposed solution. The formulation of the objective function is a critical step in defining the problem and guiding the research towards finding a solution that aligns with the goals of the study. In this study, the primary objective of the designed task scheduling method is to jointly minimize the request completion time T and overall energy consumption E for multi-task allocation. Thus, the formulation of the objective function is expressed in (18):

$$\begin{aligned} obj:& min \left\{ \omega_1 \cdot T + \omega_2 \cdot E \right\} \\ & T = max \left\{ T_{L_1}, \ldots, T_{L_V}, \ldots, T_{E_1}, \ldots, T_{E_S}, \ldots, T_{CS} \right\} \\ & E = \sum_{v=1}^{V} \sum_{k=1}^{K} E_{v,k} \end{aligned} \tag{18}$$

where ω_1 and ω_2 are weight factors, T_{L_V} and T_{E_S} denote the completion time of all tasks within the vehicle V and the edge server S, respectively. T_{CS} denotes the completion time of the tasks that are allocated to the cloud server. $E_{v,k}$ represents the consumed energy required to complete the task k generated by vehicle v.

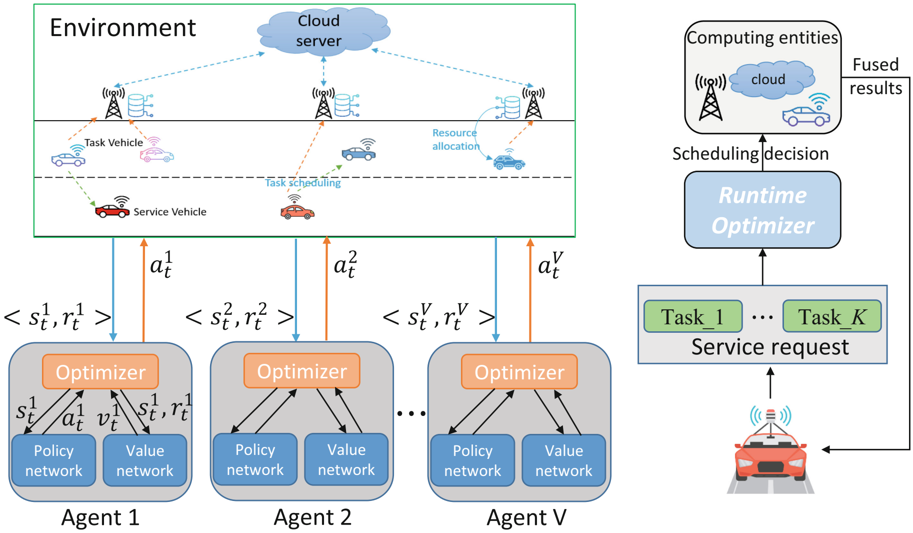


Fig. 2. Multi-agent reinforcement learning-based task scheduling procedure

4 System Design

In this section, the proposed PPO-based scheduling method is introduced.

4.1 Multi-agent Reinforcement Learning

Multi-agent reinforcement learning is a sub-field of reinforcement learning that involves multiple agents interacting with each other in a shared environment to achieve a common goal. In the context of VEC, multi-agent reinforcement learning can be used to coordinate the allocation of computing resources and the scheduling of tasks across multiple edge servers or vehicles.

In this work, as shown in Fig. 2, each vehicle can act as an agent v, which interacts with the environment and determines its task scheduling decisions, including offloading or local execution. Firstly, at each step t, the environment state s_t^v provides an observation to the policy (actor) network of each agent v. Secondly, the policy network generates the scheduling action a_t^v for v based on the observation, which is then executed in the environment. Thirdly, the environment provides reward r_{t+1}^v and new state s_{t+1}^v for each agent based on the actions taken by the agents. The value (critic) network of the agent v estimates the value of each state, which is used to calculate the advantages for policy updates. The *optimization* step updates the policy networks using backpropagation based on the advantages and the losses. The updated policies are then used to generate new trajectories, which are used for exploration and further training.

Overall, MAPPO involves multiple agents that interact with each other in a decentralized manner. The centralized training approach enables efficient learning from the collective experience of the agents, while the decentralized execution enables robust and scalable performance in complex multi-agent environments.

4.2 States, Actions, and Reward

States. The states are the present condition of the environment in which the agents operate. In our study, we depict states as a collection of observations received by vehicles from the environment. These observations encompass various aspects, such as the positions $P^e(t)$ of vehicles, the task-related information I^v from each vehicle, the transmission overhead $M^v(t)$ on each vehicle, and the computation overhead $C^e(t)$ on each computing entity. Consequently, the state space can be expressed as:

$$s_t = \{P^e(t), I^v, M^v(t), C^e(t)\} \tag{19}$$

Actions. The actions signify the choices made by agents based on the observed state of the environment. In this study, these actions encompass decisions related to offloading and the assigned transmission power for each vehicle.

$$\begin{aligned}&A_t^v = \left\{a_1^1, \ldots, a_1^K, \ldots, a_i^k, \ldots, a_V^K\right\}, \\ &a_i^k = \left\{\alpha_i^k, p_i^k\right\} \\ &\forall i \in \{1, \ldots, V\}, \forall k \in \{1, \ldots, K\}, \alpha_{i,k} \in \{0, 1, 2, 3\}\end{aligned} \tag{20}$$

where a_i^k represents the decisions made for task k generated by vehicle v, including the offloading decision α_i^k and the allocated transmission power p_i^k.

Reward. The rewards denote the responses that agents receive from the environment in accordance with their actions. They are generally crafted to incentivize agents toward actions that yield favorable outcomes, such as reducing request completion time and energy consumption.

According to the proposed objective function, the reward function is designed as:

$$r = \alpha \times \frac{exp_{time}}{max_{exp_{time}}} + \beta \times \frac{exp_{energy}}{max_{exp_{energy}}} \tag{21}$$

where exp_{time} and exp_{energy} represent the estimated completion time and energy consumption of the service request, respectively, $max_{exp_{time}}$ and $max_{exp_{energy}}$ represents the maximum exp_{time} and exp_{energy} the agent had reached. α and β are weight factors.

4.3 Deep Reinforcement Learning Algorithm

In this work, the Proximal Policy Optimisation (PPO) [14] is employed as the DRL agent training algorithm. It is a policy gradient-based actor-critic algorithm, which uses Stochastic Gradient Ascend (SGA) to update both actor (also namely policy) network π and critic (also namely value) network φ. The actor network predicts the next action and returns to the environment, while the critic network evaluates the performance of the actor network. Both networks are updated by the same loss function $L = L_{clip} + L_V + L_{entropy}$, which consists of three parts: the clipped surrogate loss L_{clip}, the value loss L_V and the entropy loss $L_{entropy}$.

In order to ensure that the policy changes are sufficiently conservative to maintain stability and avoid large deviations from the current policy, the surrogate objective function is utilized to limit the change in policy:

$$L_{clip} = \mathbb{E}[\min(\rho(s,a)\, A^{\pi_{\theta_{\text{old}}}}(s,a), \text{clip}(\rho(s,a), 1-\epsilon, 1+\epsilon)\, A^{\pi_{\theta_{\text{old}}}}(s,a))] \tag{22}$$

where the clip function clamps out-of-range values of $\rho(s,a)$ back to the constraint $(1-\epsilon, 1+\epsilon)$ instead of using KL divergence in TRPO, $\rho(s,a) = \frac{\pi_\theta(s,a)}{\pi_{\theta_{\text{old}}}(s,a)}$ is the importance sampling weight between the old $\pi_{\theta_{\text{old}}}(s,a)$ and new policy $\pi_\theta(s,a)$. Generalised Advantage Estimation (GAE) [13] is utilized to estimate the expectation of the advantage function $A^{\pi_{\theta_{\text{old}}}}(s,a)$):

$$\hat{A}_t(s,a) = \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{N-t+1}\delta_{t+1}, \tag{23}$$

where δ_t is represented by $r_t + \gamma V_{\theta_v}(s_{t+1}) - V_{\theta_v}(s_t)$, $\gamma\lambda$ is the discount factor of future rewards to control the variance of the advantage function.

The prediction of values V indicates the ability to predict accumulated rewards. Therefore, it should be close to the advantage at a specific pair of the state s and action a, which formulates the equation of the value loss L_V:

$$L_V = \mathbb{E}_{s,a}[A^{\varphi}(s,a)] \tag{24}$$

where we use the Mean Squared Error (MSE) to estimate the expectation and minimise the loss.

Lastly, the entropy loss $L_{entropy}$ is introduced to encourage exploration during the learning process, which is added as a regularisation term. This is calculated by the negative entropy of the policy distribution, involving both continuous and discrete actions. As a result, the DRL agent may be less sensitive to local optimal solutions, which can lead to a more robust policy.

5 Experiment

5.1 Experimental Setting

Training and Test Environments. In the experimental phase, the multi-agent reinforcement learning-based task scheduling models are trained on a server equipped with an Intel Xeon W-22555 processor and an NVIDIA RTX 3070 GPU. Subsequently, the experiments are performed on a Windows 11 PC featuring an Intel i7-13700H processor and 16 GB DRAM. The detailed experimental parameters are outlined in Table 1.

Table 1. Experimental parameter setting

Parameters	Value	Parameters	Value
Number of vehicles	4–10	Data size	2–10 Mb
Number of edge servers	8	Bandwidth of edge server	100 MHz
Number of tasks	3–7	Speed of vehicles	≈ 15 m/s
Required CPU cycles of the tasks	10–20 cpu cycles	Transmission power of vehicles	1 dBm
Computation power of the cloud	4 cpu cycles/s	Execution power of vehicles	3-4 dBm
Computation power of edge servers	2 cpu cycles/s	Computation power of vehicles	1 cpu cycles/s

5.2 Benchmark Methods

In the experiments, the proposed PPO-based scheduling method is compared with the following methods:

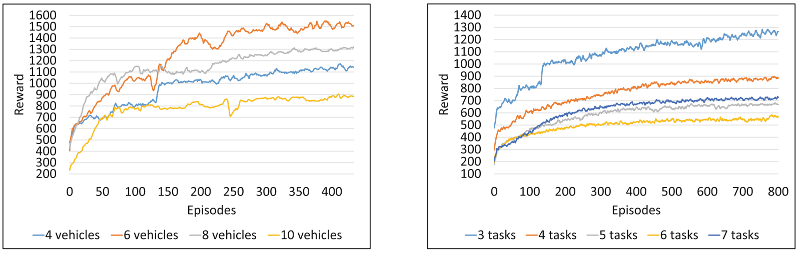
(1) **Deep Deterministic Policy Gradient (DDPG)** [15] The task scheduling method based on DDPG designed in [15] primarily focuses on optimizing the request completion time. (2) **Random Scheduling (RS)** The RS

method makes task scheduling decisions randomly, by which the tasks are executed locally or distributed to edge servers and other vehicles randomly. (3) **Offloading-only Scheduling (OS)** The OS method allocates all tasks exclusively to edge servers or other vehicles. (4) **Local-only Scheduling (LS)** Different from the OS method, the LS method involves the execution of all tasks locally.

5.3 Performance Evaluation

Convergence. The convergence of the reward in multi-agent reinforcement learning refers to the process by which the rewards earned by the agents over time become more stable and consistent. Achieving convergence is an important goal in multi-agent reinforcement learning as it indicates that the agents have reached a level of performance that is close to optimal and can be used to make decisions in the real world. In this work, convergence is essential to ensure that the learning process terminates and the algorithm can produce a policy that performs well in making task scheduling decisions in the given environment.

The rewards are shown in Fig. 3a and Fig. 3b, where x coordinate denotes the current training episode and y coordinate denotes the reward obtained from the scheduling decision. As can be seen from the figures, the proposed PPO-based method converges fast under different numbers of vehicles and tasks.

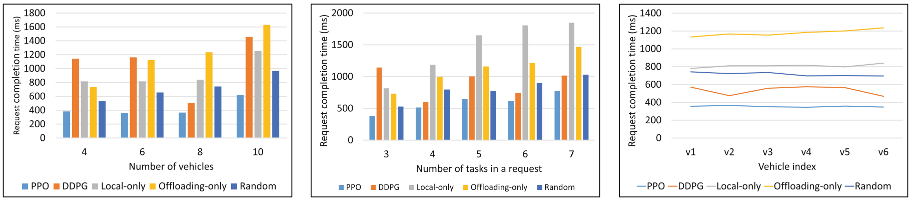


(a) Different number of vehicles (b) Different number of tasks

Fig. 3. Convergence property

Request Completion Time. The service request completion time in vehicular edge computing refers to the time taken to complete a service request from the time it is generated by the vehicle until the result is feedback to the user. It is an important performance metric that determines the quality of service (QoS) in vehicular edge computing. The service request completion time can be influenced by several factors, mainly including the task processing time and the transmission delay between the vehicles and edge servers.

Figure 4a shows that the PPO method achieves shorter request completion times compared to other methods, for different numbers of vehicles. Figure 4b illustrates that as the number of tasks within a request increases, the completion

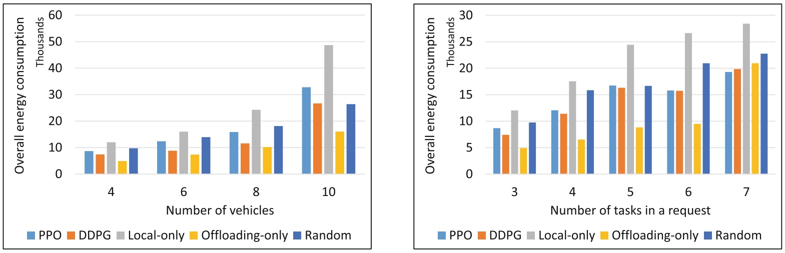


(a) Different number of vehicles (b) Different number of tasks (c) Completion time of each vehicle

Fig. 4. Comparison of the request completion time

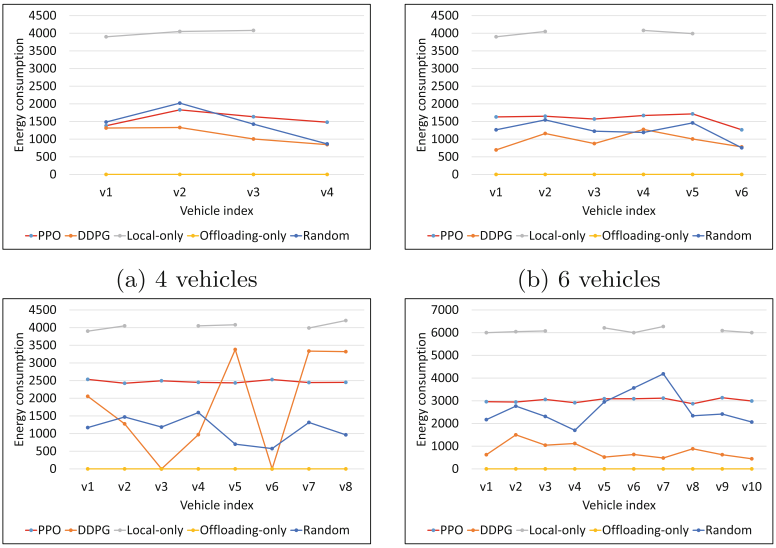
time also increases, however, the PPO-based scheduling method still achieves the shortest request completion time. In Fig. 4c, the request completion time for each vehicle is presented, and it can be observed that the PPO-based scheduling method results in similar completion times for requests generated from all vehicles.

Overall Energy Consumption. Overall energy consumption refers to the total electrical energy required to complete all the tasks in one request, including the energy consumption on task execution and data transmission involved in delivering service requests. Figure 5 compares the overall energy consumption under different conditions. As shown in Fig. 5a and Fig. 5b, vehicles have less energy consumption when all the tasks are offloaded based on the offloading-only method. The energy consumption based on the PPO method is similar to the DDPG method but lower than the local-only method and random-based method. Because all the tasks are executed locally based on a local-only scheduling method can result in significantly higher energy consumption compared to offloading them rationally.



(a) Different number of vehicles (b) Different number of tasks

Fig. 5. Comparison of the overall energy consumption



(a) 4 vehicles

(b) 6 vehicles

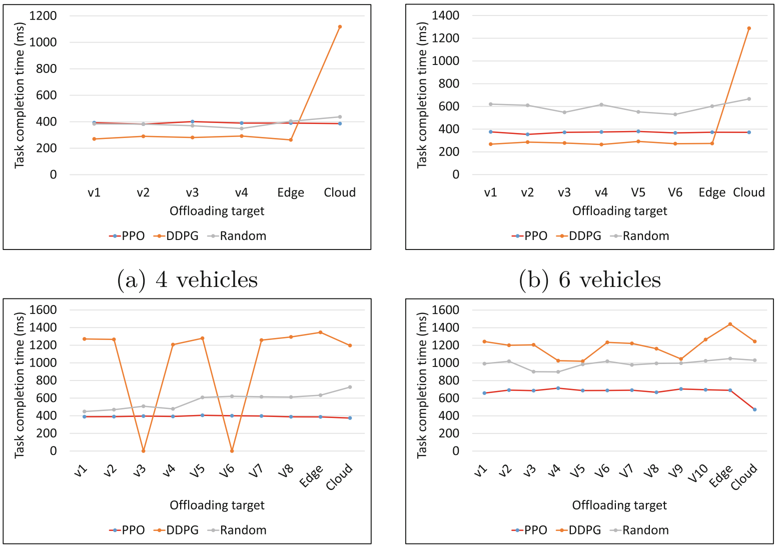
(c) 8 vehicles

(d) 10 vehicles

Fig. 6. Energy consumption on each vehicle

Energy Consumption on Each Vehicle. Balanced energy consumption can prevent overloading of a single vehicle. In a multi-vehicle system, if one vehicle consumes significantly more energy than others, it can become overloaded, causing delays in service delivery and reduced system performance. Figure 6 compares the energy consumed by each vehicle when processing tasks that are offloaded to it and generated by itself. When there are four vehicles, $v4$ served as the Service Vehicle (SeV), which does not generate task processing requests. Similarly, $v3$ and $v6$ are SeVs when there are six and eight vehicles. $v4$ and $v8$ are SeVs when there are ten vehicles. Other vehicles are called Task Vehicles, which can generate task processing requests. As shown in Fig. 6, the energy consumption on each vehicle, no matter the SeV or TaV, is almost the same when the tasks are scheduled based on the PPO method. In contrast, the energy consumption is not distributed equally among all the vehicles when using the DDPG method and random scheduling.

Task Completion Time on Each Computing Entity. The task completion time on each entity refers to the completion time of all the tasks that are allocated to the computing entity. Figure 7 illustrates a comparison of the task completion time among all computing entities, including vehicles, edge servers, and the cloud server. Based on the PPO scheduling method, the tasks allocated to all the entities are completed at almost the same time. However, based on the DDPG scheduling method, the completion time of the cloud server is much



(a) 4 vehicles

(b) 6 vehicles

(c) 8 vehicles

(d) 10 vehicles

Fig. 7. Task completion time on each computing entity

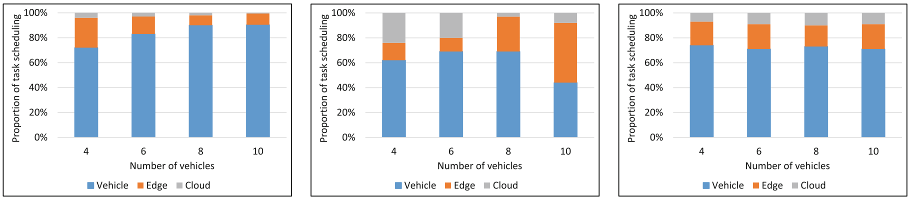
higher than other computing entities when the number of vehicles is small, ultimately affecting the completion time of the vehicles' service requests.

Proportion of Task Scheduling. In Fig. 8 and Fig. 9, the proportion of the tasks that offloaded to vehicles, edge server, and the cloud server are compared.

As shown in Fig. 8a, with the number of vehicles increasing, more tasks are executed on vehicles and the proportion of tasks offloaded to edge servers or the cloud server decreases because the increasing number of vehicles can provide more computing resources, allowing more tasks to be executed on vehicles. However, in Fig. 8b, there is no clear trend in the change of proportion when scheduling tasks based on the DDPG method. In addition, the proportion of the offloaded tasks remains constant when using a random-based scheduling method, as shown in Fig. 8c.

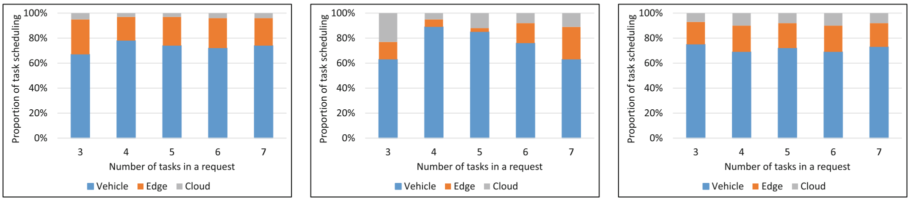
As shown in Fig. 9a and Fig. 9c, the scheduling proportions of the tasks based on the PPO-based and random-based methods are not affected by the number of tasks in one service request. However, the PPO-based scheduling method offloads fewer tasks to the cloud server, which efficiently alleviates the bandwidth pressure for vehicles.

Quality of Method. As the goal of this study is to improve both the completion time and energy consumption of requests, a metric is needed to compare the overall performance of different methods in achieving these objectives. To this end, the Quality of Method (QoM) is defined and used to quantify and evaluate



(a) PPO method (b) DDPG method (c) Random method

Fig. 8. Proportion of task scheduling under different numbers of vehicles



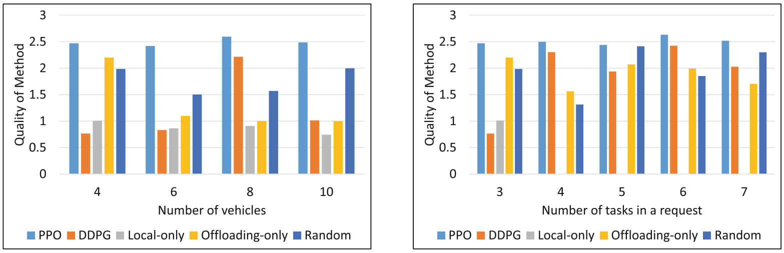
(a) PPO method (b) DDPG method (c) Random method

Fig. 9. Proportion of task scheduling under different numbers of tasks in a request

the performance of the method in a comprehensive manner. The QoM of each method can be calculated by:

$$QoM_i = \mu_1 \frac{T_{max} - T_i}{T_{max} - T_{min}} + \mu_2 \frac{E_{max} - E_i}{E_{max} - E_{min}} \tag{25}$$

where μ_1 and μ_2 represent the weight value of completion time and energy consumption, respectively. T_i and E_i represent the completion time and energy consumption of method i, respectively. In this work, we set $\mu_1 = 2$ and $\mu_2 = 1$. As shown in Fig. 10, the QoM of the PPO-based scheduling method is much higher than other methods no matter the number of vehicles and tasks. This means that the proposed PPO-based scheduling method can always have a better comprehensive performance in terms of shortening request completion time and reducing energy consumption.



(a) Different number of vehicles (b) Different number of tasks

Fig. 10. Comparison of the Quality of Method

6 Conclusion

In this work, a collaborative multi-task scheduling strategy is designed utilizing the multi-agent Proximal Policy Optimization algorithm and cloud-edge-end collaboration. The proposed method allows multiple computation tasks in one service request to be either executed locally or remotely by offloading to edge servers and the cloud server simultaneously, which helps to decrease request completion time. Moreover, the transmission power can be dynamically adjusted based on the distances between vehicles and other computing entities, which helps to reduce energy consumption. The experimental results demonstrate that the PPO-based scheduling method outperforms other methods in terms of reducing request completion time. Additionally, it achieves balanced energy consumption and task workload distribution among all computing entities, preventing overloading of a single entity. Furthermore, allocating tasks based on the PPO-based method results in fewer tasks being offloaded to the cloud server, alleviating the bandwidth pressure for vehicles. Finally, the comprehensive performance evaluation QoM results show that the proposed method outperforms other methods.

Acknowledgments. This research was funded by: Key Program Special Fund in XJTLU under project KSF-E-64; XJTLU Research Development Funding under projects RDF-19-01-14 and RDF-20-01-15; the National Natural Science Foundation of China (NSFC) under grant 52175030.

References

1. Abdelkader, G., Elgazzar, K., Khamis, A.: Connected vehicles: technology review, state of the art, challenges and opportunities. Sensors **21**(22), 7712 (2021)
2. Alqarni, M.A., Mousa, M.H., Hussein, M.K.: Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing. J. King Saud Univ. Comput. Inf. Sci. **34**(10), 10356–10364 (2022)
3. Fan, W., Liu, J., Hua, M., Wu, F., Liu, Y.: Joint task offloading and resource allocation for multi-access edge computing assisted by parked and moving vehicles. IEEE Trans. Veh. Technol. **71**(5), 5314–5330 (2022)
4. Huang, J., Wan, J., Lv, B., Ye, Q., Chen, Y.: Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Syst. J. **17**, 2500–2511 (2023)
5. Ju, Y., et al.: Joint secure offloading and resource allocation for vehicular edge computing network: a multi-agent deep reinforcement learning approach. IEEE Trans. Intell. Transp. Syst. **24**, 5555–5569 (2023)
6. Kazmi, S.A., Otoum, S., Hussain, R., Mouftah, H.T.: A novel deep reinforcement learning-based approach for task-offloading in vehicular networks. In: 2021 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2021)
7. Li, P., Wang, X., Huang, K., Huang, Y., Li, S., Iqbal, M.: Multi-model running latency optimization in an edge computing paradigm. Sensors **22**(16), 6097 (2022)
8. Liu, L., Feng, J., Mu, X., Pei, Q., Lan, D., Xiao, M.: Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Trans. Intell. Transp. Syst. **24**, 15513–15526 (2023)

9. Lu, S., Shi, W.: Vehicle computing: vision and challenges. J. Inf. Intell. **1**, 23–35 (2022)
10. Raza, S., Wang, S., Ahmed, M., Anwar, M.R., et al.: A survey on vehicular edge computing: architecture, applications, technical issues, and future directions. Wirel. Commun. Mob. Comput. **2019**, 1–19 (2019)
11. Rubio-Loyola, J., et al.: Towards intelligent tuning of frequency and transmission power adjustment in beacon-based ad-hoc networks. In: VEHITS, pp. 648–656 (2018)
12. Saleem, U., Liu, Y., Jangsher, S., Li, Y., Jiang, T.: Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing. IEEE Trans. Wireless Commun. **20**(1), 360–374 (2020)
13. Schulman, J., Moritz, P., Levine, S., Jordan, M., Abbeel, P.: High-dimensional continuous control using generalized advantage estimation. arXiv preprint arXiv:1506.02438 (2015)
14. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
15. Tian, H., et al.: CoPace: edge computation offloading and caching for self-driving with deep reinforcement learning. IEEE Trans. Veh. Technol. **70**(12), 13281–13293 (2021)
16. Wang, H.: Collaborative task offloading strategy of UAV cluster using improved genetic algorithm in mobile edge computing. J. Robot. **2021**, 1–9 (2021)
17. Waqar, N., Hassan, S.A., Mahmood, A., Dev, K., Do, D.T., Gidlund, M.: Computation offloading and resource allocation in MEC-enabled integrated aerial-terrestrial vehicular networks: a reinforcement learning approach. IEEE Trans. Intell. Transp. Syst. **23**(11), 21478–21491 (2022)
18. Xu, X., et al.: Joint task offloading and resource optimization in NOMA-based vehicular edge computing: a game-theoretic DRL approach. J. Syst. Architect. **134**, 102780 (2023)