Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Pressure Guided Hierarchical Offloading and Heterogeneous Resource Selection for UAV-assisted Vehicular Edge Computing Networks: An Perspective of Future Task Prediction

Chao He

hechao@sanxiau.edu.cn

Chongqing Sanxia University of Science and Technology

Dongfeng Fu

Chongqing Sanxia University of Science and Technology

Xin Xie

Chongqing University of Posts and Telecommunications

Jinkui Zhang

Chongqing Changan Science and Technology

ShuXu Yue

Yunnan Water Resources and Hydropower Vocational College

Zheng Zhang

Brunel University of London

Research Article

Keywords: vehicular edge computing networks, task offloading, future task pressure, hierarchical decision, heterogeneous edge resources

Posted Date: October 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11334702/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Pressure Guided Hierarchical Offloading and Heterogeneous Resource Selection for UAV-assisted Vehicular Edge Computing Networks: An Perspective of Future Task Prediction

Chao He[1*], Dongfeng Fu[1], Xin Xie[2], Jinkui Zhang[3], ShuXu Yue[4*], Zheng Zhang[5]

[1]School of Electronic and Information Engineering, Chongqing Sanxia University of Science and Technology, Chongqing, 404120, China.
[2]School of Automation, Chongqing University of Posts and Telecommunications, Chongqing, 400065, China.
[3]Department of Artificial Intelligence Foundations and Applications, Chongqing Changan Science and Technology, Chongqing, 401133, China.
[4]School of Electrical Engineering, Yunnan Water Resources and Hydropower Vocational College, Kunming, 650499, China.
[5]Department of Computer Science, Brunel University London, Uxbridge, Middlesex, UB8 3PH, United Kingdom.

*Corresponding author(s). E-mail(s): hechao@sanxiau.edu.cn; yueshuxu@ynwater.edu.cn;
Contributing authors: 17320370160@163.com; xiexin@cqupt.edu.cn; zhangjk3@changan.com.cn; Zheng.Zhang2@brunel.ac.uk;

Abstract

Vehicular edge computing can combine local vehicles, cooperative vehicles, unmanned aerial vehicles (UAVs), and roadside units, but current observations do not fully describe the task pressure that may appear in later slots. This paper presents Pressure Guided Hierarchical Offloading (PGHO), which summarizes predicted workload, task arrivals, urgency, and deadline changes as future task pressure before heterogeneous resource selection in UAV-assisted vehicular edge computing networks. The upper policy selects an execution node using current resource conditions together with this future information, and the lower policy then controls UAV movement and the offloading ratio under the link, queue, and

resource conditions of the selected node. Across normal load, high load, and tight deadline scenarios, PGHO achieves 0.4301 s average delay and 99.705% task success ratio. Compared with multiagent deep deterministic policy gradient, soft actor critic, and hybrid multiagent proximal policy optimization, average delay is reduced by 21.1%, 12.1%, and 7.4%, respectively. PGHO also keeps average delay close to task flow greedy scheduling while improving task success ratio by about 2.62 percentage points. Component removal experiments show that future task information and hierarchical decision making mainly reduce average and tail delay, while action masking supports reliability under tight deadlines. The results indicate that PGHO maintains low delay and high task completion across different task pressure conditions.

Keywords: vehicular edge computing networks, task offloading, future task pressure, hierarchical decision, heterogeneous edge resources

MSC Classification: 68M14, 68M20, 68T05

1 Introduction

Internet of Vehicles (IoV) is evolving from a traditional information exchange network into a mobile intelligent system that combines sensing, computing, communication, and coordinated control. Autonomous vehicles must continuously process tasks such as object recognition, environment understanding, trajectory inference, road coordination, and safety decisions. These tasks often have large input sizes, high computational density, continuous arrivals, and different urgency levels. Onboard terminals are limited by processing capacity, storage, and energy. Under dense traffic or bursty tasks, local computing alone easily causes queue buildup and longer service delay. A recent survey of vehicular edge computing describes how computing resources near vehicles can support task offloading and relieve onboard resource limitations [1]. Vehicular edge computing (VEC) further includes nearby vehicles, roadside infrastructure, and edge servers in one cooperative system, providing a basis for task offloading in dynamic road environments [2]. Recent VEC work has also combined adaptive traffic state assessment with federated reinforcement learning to reduce energy consumption and task latency under changing traffic conditions [3].

Fixed roadside units (RSUs) can provide stable computing resources, but their coverage and deployment density are limited by road space, construction cost, and local traffic distribution. When vehicles gather in a hot spot, a fixed node may face communication congestion, longer queues, and insufficient processing capacity at the same time. An unmanned aerial vehicle (UAV) changes this situation by bringing a movable radio and computing point into the same area. The source vehicle, cooperative vehicles, UAVs, and RSUs can then form a heterogeneous resource pool spanning aerial and ground resources. Prior studies treat trajectory, link state, computation allocation, and energy constraints as coupled decisions in edge computing with UAV support [4]. Vehicular studies also report that changing aerial position can recover reachability where roadside coverage is sparse [5]. Task offloading with UAV support

has also been integrated with digital twin and generative artificial intelligence (AI) for joint task offloading and resource allocation in IoV [6]. The resulting resource pool is not simply larger; which nodes are usable, and how long they remain useful, now varies with both road traffic and UAV motion.

Heterogeneity between aerial and ground resources expands the set of possible execution nodes, but it also makes offloading decisions harder. Vehicle motion continuously changes node distance, reliable transmission rate, and connection duration. Task arrivals, computational load, urgency, and deadlines vary with traffic density, sensing cycles, and local events. A decision made in the current slot determines where the present task is transmitted and processed, while also changing later queue buildup, residual node capacity, and UAV position. Conventional static optimization depends on accurate models and prior distributions, and repeated solution becomes impractical under high-dimensional states, coupling across slots, and real-time requirements. Deep reinforcement learning can learn long-horizon policies through continued interaction with a dynamic environment and is now widely used for edge offloading and vehicular resource allocation [7]. Deep reinforcement learning has also been combined with a Nash equilibrium game for IoV task offloading optimization [8]. Generative AI has been combined with fifth generation mobile networks to improve computation offloading for complex IoV tasks [9]. Even when predictive information is available, a key remaining question is how future task changes should alter heterogeneous resource selection before they become visible in the current queue.

Consider two slots with almost the same queue length and radio quality. The first is followed by ordinary arrivals, while the second is followed by a short burst of urgent tasks. A policy that sees only the current snapshot has little reason to choose differently, even though the second choice can create a substantially heavier queue a few slots later. Prediction can move service decisions earlier in time, but the issue considered here is not simply whether future information should be used. The more specific question is how several future task attributes can be represented together before heterogeneous resource selection, and how that information can remain relevant after the execution resource has been chosen.

The forecast still has to be translated into a choice among heterogeneous resources. More computing capacity does not automatically make a candidate the better execution place. Local processing avoids transmission but can be slow. A cooperative vehicle may offer spare cycles only while the contact lasts. UAV service depends on where the aircraft can move and on the link available from that position. An RSU may be powerful and still carry the longest queue in a crowded area. These quantities have to be considered together rather than through one rule for each node class. Studies of hybrid VEC systems consider computing, communication, and resource conditions jointly [10]. The resource description should therefore preserve the differences among node types while allowing the feasible set and the expected burden of each candidate to change with the slot.

Once the candidate resources can be compared, the control variables introduce a second difficulty. The execution node is a discrete choice, whereas UAV displacement and the offloading ratio are continuous. Changing the node immediately changes the link used, the queue that receives the task, and the useful range of the ratio;

UAV movement then changes later coverage. Treating all of these variables as one flat action makes the policy search over many combinations that are physically weak or mutually inconsistent. Multi-agent control can coordinate UAV motion [11], while hierarchical reinforcement learning provides a way to decompose complex decisions into subtasks [12]. In Pressure Guided Hierarchical Offloading (PGHO), the levels are organized around the dependency between the decisions: the selected heterogeneous execution resource determines the link, queue, resource availability, and ratio conditions under which the following continuous action is generated.

These two decision issues motivate the online decision components of PGHO. Future task pressure combines several types of expected task variation before the upper resource decision, rather than treating one predicted quantity as the sole description of future demand. The node selected by the upper policy then supplies the link, queue, resource availability, and ratio conditions used by the lower controller. PGHO therefore connects future task evolution with heterogeneous resource selection and then carries the selected resource conditions into continuous control.

A separate issue during training concerns which learned policy should be retained after training: the final network update is not necessarily the policy with the best validation performance on a common task flow. Evaluation can still fluctuate because of initialization, exploration, replay data, and simultaneous updates among agents. To address this issue, PGHO compares saved policies on the same validation flow and retains the candidate with the best observed validation performance. Checkpoint storage and restoration support the selection of policies that may perform better on validation than the policy obtained at the end of training. The main contributions are summarized as follows:

(1) A future task pressure representation jointly summarizes predicted workload, task arrivals, urgency, and deadline changes before resource selection. This representation makes several types of future task evolution available to the upper decision together with the current conditions of heterogeneous resources.

(2) A hierarchical decision structure is organized around the dependency between resource selection and subsequent continuous control. The selected execution node supplies the link, queue, resource availability, and feasible ratio conditions used by the lower policy for UAV movement and offloading control.

(3) Policy selection on a fixed validation flow is used during training to retain policies with better validation performance. Elite experience and rollback remain auxiliary operations within this procedure.

The rest of this paper is structured as follows: Section 2 reviews heterogeneous resource organization, use of future task information, hierarchical decision mechanisms, and policy evaluation and selection during training. Section 3 gives the heterogeneous resource model with aerial and ground nodes, task execution model, joint decision model, and problem formulation. Section 4 presents the future task pressure representation, hierarchical decision making with predicted task pressure, and policy selection using validation results. Section 5 gives the experimental settings, main method comparison, and component removal results. Section 6 summarizes the conclusions and limitations.

2 Related Work

Task offloading in VEC with UAV support involves candidate resource organization, future task information, discrete resource selection, continuous control, and final policy evaluation. Predicted task demand can affect resource selection, while the selected node determines the conditions for subsequent continuous control.

This section reviews heterogeneous resource organization, the use of future task information, and hierarchical decision structures. The last part discusses how validation results are used for final policy selection.

2.1 Heterogeneous Resource Pooling

VEC usually treats the source vehicle, nearby vehicles, RSUs, and edge servers as candidate execution nodes. Rahmani et al. [13] reviewed offloading for autonomous vehicles across edge computing architectures, optimization approaches, intelligent algorithms, communication designs, and security concerns. Local computing does not require wireless transmission but has limited processing capacity. Nearby vehicles can share idle resources, but mobility and contact duration limit their service. Roadside nodes are more stable, but coverage and queue conditions depend on deployment location and traffic density. These forms of cooperation enlarge the candidate set. Liqun Yang [14] investigated task offloading and resource scheduling among vehicles, RSUs, and cloud resources, using vehicle clustering and a Selfish Herd Optimizer for task scheduling. These studies provide a basis for heterogeneous resource selection, but different node types often use different state structures or action encodings. A common method for comparing capacity, queue, distance, and reliability still needs to be made clear.

A UAV changes the structure of the candidate resources. The UAV can move closer to the task source, improve edge coverage, and process part of the task, but its availability is also affected by position, flight region, link quality, and movement cost. Sun et al. [15] analyzed UAV resource scheduling in an IoV setting. Jeong et al. [16] modeled aerial edge computing through bit allocation and path planning. These scheduling and path planning studies illustrate how UAV mobility and computation decisions interact. Existing studies show the value of aerial nodes, but many models define variables for specific node types and lack a unified representation for local vehicles, cooperative vehicles, UAVs, and RSUs.

Long et al. [17] investigated how task splitting and computation resource allocation can be jointly optimized to balance latency and energy consumption in VEC. From the decision perspective, a unified resource representation must deal with both feasibility and comparability. Feasibility asks whether a node can be selected under the current distance, rate, and transmission time conditions, while comparability asks which feasible node is more suitable when computing, queue, and link conditions change together. Existing work therefore motivates a common candidate representation that retains the physical differences among local vehicles, cooperative vehicles, UAVs, and RSUs.

2.2 Use of Future Task Information

In the considered setting, task attributes can change over short intervals. Sensing cycles, local traffic, road events, and service priority can change both the work arriving at the system and the time allowed for completion. Related scheduling work in multisensor industrial Internet of Things systems shows that time-sensitive information freshness can depend on scheduling under shared server resources [18]. This reinforces the importance of making resource decisions before time-sensitive conditions deteriorate. Predictive offloading has used vehicle motion and connection changes to act earlier [19], while trajectory prediction has been used to reduce offloading delay in VEC systems [20]. These approaches mainly provide future mobility or connectivity information.

Future information has also been introduced from the task and traffic side. Wu et al. [21] combined task demand prediction with UAV edge offloading and energy optimization. Chhabra et al. [22] used traffic information in collaborative task offloading and service caching. These studies demonstrate that prediction can directly support resource decisions rather than serving only as an auxiliary observation. The remaining issue considered in this paper is more specific: workload, task arrivals, urgency, and deadline pressure can evolve together, and their combined effect has to be represented before a heterogeneous execution resource is selected. PGHO addresses this issue by forming future task pressure from several predicted task attributes and carrying that combined information into the resource decision.

2.3 Hierarchical Decisions with Discrete and Continuous Actions

Deep reinforcement learning has been widely used to adapt VEC offloading decisions to changing network and resource states. Bo and Zhao [23] developed a multiagent reinforcement learning strategy for vehicular task offloading under resource constraints. Sun et al. [24] combined long short term memory prediction with multiagent reinforcement learning for computation offloading in vehicular networks, while Luo et al. [25] studied self-learning computation offloading. Yao et al. [26] further addressed a dynamic edge setting. Wu et al. [27] combined twin delayed deep deterministic policy gradient with a server selection procedure to address task offloading and load balancing in VEC. More generally, Yang et al. [28] investigated how deep reinforcement learning can adapt offloading decisions to fluctuating wireless channels and uncertain task arrivals in mobile edge computing. The remaining difficulty in the system considered here is that execution location, UAV movement, and offloading ratio do not have the same action form.

This distinction is also visible in systems with UAV support. Peng et al. [29] studied resource management in VEC systems with UAV support. Zhao et al. [30] used multi-agent reinforcement learning for offloading with UAV support. Ju et al. [31] jointly considered offloading and path optimization. Liu et al. [32] modeled random task arrivals, queue evolution, and UAV motion, using Lyapunov optimization with deep deterministic policy gradient (DDPG) to reduce transmission energy while maintaining queue stability. Multiagent deep deterministic policy gradient (MADDPG) is

a common controller when the main decisions are continuous [33]. Zakaryia et al. [34] optimized task allocation, UAV trajectories, computing resources, and communication resources in a system with multiple UAVs and mobile edge computing (MEC). These methods motivate coordinated control, but a discrete execution node is still difficult to represent as a continuous preference when the set of feasible candidates changes with the system state.

Recent hierarchical VEC studies provide a closer comparison. Liu et al. [35] used an upper double deep Q network (DDQN) for discrete communication decisions and a lower DDPG for continuous power, offloading, and resource allocation decisions. Du et al. [36] separated service deployment and replacement from continuous task offloading ratios through high and low decision levels. Yang and Yoo [37] organized UAV deployment and collaborative task offloading over two time scales in a hierarchical reinforcement learning framework. Navidi et al. [38] combined learning automata for strategic policy selection with a deep recurrent Q network for control of offloading ratios in IoV. These studies organize decision making across different action types or time scales. PGHO organizes the hierarchy around the dependency between heterogeneous execution resource selection and the continuous UAV control that follows, while several predicted task attributes are made available before the resource decision.

2.4 Policy Evaluation and Selection During Training

Training introduces a separate issue from online action generation: which saved policy should be retained after learning. Late training behaviour can depend on initialization, exploration, replay data, and the timing of agent updates. Henderson et al. [39] showed why fixed evaluation protocols and independent runs matter for reproducibility. Work on self-imitation also suggests that useful historical trajectories can help retain good behaviour [40]. These observations motivate evaluating retained policies under common validation conditions rather than assuming that the final training update is always the most suitable policy for later evaluation.

Taken together, existing studies show that prediction can support offloading decisions, hierarchy can separate discrete and continuous control, and validation can support policy retention during learning. PGHO focuses primarily on the connection between future task information and hierarchical control under the conditions of the selected resource, while validation results are used during training to select the retained policy.

3 System Model and Problem Formulation

Figure 1 illustrates the heterogeneous resource pooling and task offloading scenario considered in this paper. The scenario model describes the source vehicle, cooperative vehicles, UAVs, and RSUs as candidate execution resources and retains their computing, queue, coverage, position, link, and energy conditions. The execution model describes task transmission, local and remote computation, parallel execution, completion delay, and task failure. The decision model defines the state information and the joint choice of execution node, UAV movement, and offloading ratio.

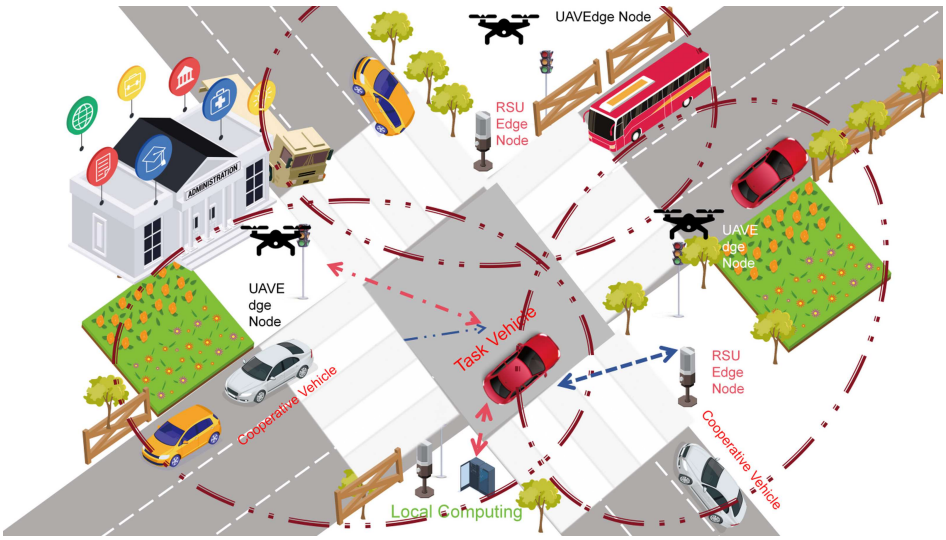


Fig. 1 Heterogeneous resource pooling and task offloading in VEC with UAV support

3.1 Scenario Model

We consider dynamic task processing in VEC with UAV support and construct the heterogeneous resource architecture with aerial and ground nodes shown in Fig. 2. Unlike formulations that introduce separate decision variables for each node type, the model places local vehicles, cooperative vehicles, UAVs, and RSUs in one candidate execution space. Different resources can therefore be represented and compared within the same decision space while retaining the physical conditions of each node.

Figure 2 shows how the four resource types enter one candidate pool while retaining different mobility, coverage, queue, and computing conditions. This common representation is used only for resource comparison and does not remove physical constraints specific to each node.

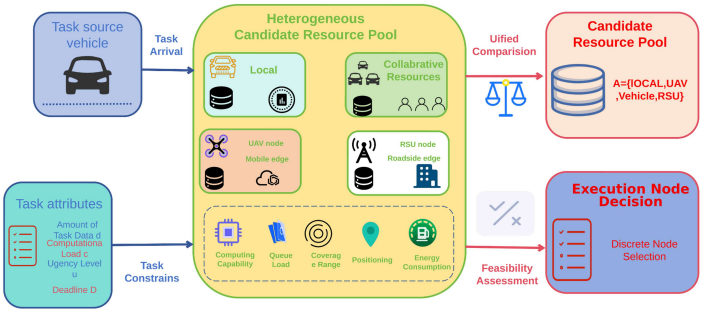


Fig. 2 Heterogeneous resource pool with aerial and ground nodes

The system contains task source vehicles, cooperative vehicles, UAVs, and RSUs. A candidate keeps its own computing capacity, queue, coverage, position, link reliability, and normalized energy coefficient. The same state representation is used for all node types, while their physical values remain different. Local, cooperative, aerial, and roadside execution can therefore be represented without defining a separate decision structure for each node class.

Task flow and resource state evolution are described in discrete time slots. Each slot contains task arrival, state construction, joint decision making, task execution, and queue update. The current action changes node queues, UAV positions, and later resource availability, so slots are not independent. In each slot, a source vehicle generates a task whose input size, computational density, urgency, and deadline are represented by the attribute vector

$$\tau_{i,t} = (d_{i,t}, c_{i,t}, u_{i,t}, D_{i,t}) \tag{1}$$

The four entries of this vector are used in the computation, deadline, and pressure models. Specifically, $d_{i,t}$, $c_{i,t}$, $u_{i,t}$, and $D_{i,t}$ denote the task input size, computational density, task urgency, and deadline, respectively; their units are Mbit, Mcycles/Mbit, dimensionless, and seconds.

Transmission rate is measured in Mbit/s. Transmission delay can therefore be calculated directly from the offloaded data and the link rate, while the required computation follows from input size and computational density. This paper adopts a divisible task model. Task characteristics vary with the load state, burst state, and service urgency described in the scenarios below. For a divisible task, the total number of computational cycles can be calculated as

$$C_{i,t} = d_{i,t} c_{i,t} \tag{2}$$

The product in Eq. (2) provides the workload later divided between local and remote execution.

Task parameters are not generated independently. They are driven by a temporally correlated task flow state. Computational load, arrival intensity, urgency, and deadlines evolve with the underlying task flow and are adjusted by a common set of rules for each scenario. The normal scenario represents ordinary pressure. The high load scenario increases input size and queue competition, while the tight deadline scenario shortens task deadlines. Scenario differences therefore arise from combinations of task pressure rather than from a single isolated parameter.

The offloading ratio satisfies $0 \leq \rho_{i,t} \leq 1$, and the local and remote workloads are given by $C_{i,t}^{\mathrm{loc}} = (1 - \rho_{i,t}) C_{i,t}$ and $C_{i,t}^{\mathrm{off}} = \rho_{i,t} C_{i,t}$, respectively.

When $\rho_{i,t} = 0$, the task is executed entirely on the source vehicle. When $0 < \rho_{i,t} < 1$, the task is divided between local and remote execution. When $\rho_{i,t} = 1$, the entire task is executed at the selected remote node.

A task may be divided proportionally between local and remote processing, and the two parts execute in parallel. This preserves the immediacy of local processing while allowing the policy to use the greater capacity of a remote node. This is a modeling assumption of the present system.

To describe heterogeneous service nodes using one model, each candidate in the resource pool is represented by multiple attributes. The candidate index identifies its position in the common candidate set, while node type, queue, computing capacity, and communication attributes remain in the state. The policy therefore does not require a different selection procedure for each node type. Accordingly, the candidate execution space is expressed as

$$\mathcal{M} = \left\{ s_i^{loc} \right\} \cup \mathcal{U} \cup \mathcal{C} \cup \mathcal{R} \tag{3}$$

The union in Eq. (3) includes the local vehicle, UAVs, cooperative vehicles, and RSUs without imposing identical resource conditions.

The resource nodes and time slots are indexed by $m \in \mathcal{M}$ and $t \in \mathcal{T}$, respectively. The resource state of node m in slot t is defined as

$$\mathbf{z}_{m,t} = (f_m, Q_{m,t}, R_m, P_m, \kappa_m, \mathbf{p}_{m,t}) \tag{4}$$

In Eq. (4), f_m, $Q_{m,t}$, R_m, P_m, κ_m, and $\mathbf{p}_{m,t}$ denote the computing capacity, queued computational workload, communication coverage range, transmit power, computing energy coefficient, and position of node m, respectively. The state keeps these quantities specific to each node so that heterogeneous candidates remain comparable without being treated as physically identical.

UAV positions vary over time. We control horizontal UAV motion in two dimensions at a fixed altitude, consistent with path planning and task allocation models for UAV computing nodes. Movement changes the distance between a source vehicle and an aerial node, and therefore changes reliable transmission rate, coverage, and remote completion delay. UAV movement must consequently be considered together with node selection and the offloading ratio. The next UAV position is obtained by adding the displacement and projecting the result onto the permitted flight region:

$$\mathbf{p}_{u_k,t+1} = \mathrm{Proj}_{\Omega_u} \left(\mathbf{p}_{u_k,t} + \Delta \mathbf{p}_{u_k,t} \right) \tag{5}$$

The projection in Eq. (5) prevents the UAV from leaving the permitted region. The maximum horizontal speed further imposes the displacement bound

$$\| \Delta \mathbf{p}_{u_k,t} \|_2 \leq v_{\max} \Delta t \tag{6}$$

The projection operator $\mathrm{Proj}_{\Omega_u}(\mathbf{z}) = \arg\min_{\mathbf{p} \in \Omega_u} \|\mathbf{p} - \mathbf{z}\|_2$ confines the updated position to the permitted flight region, while Eq. (6) limits the displacement within one slot by the maximum horizontal speed.

To represent node load under competition among tasks, a discrete queue model is used. Let the new computational workload assigned in slot t to node m be A. Accounting for the workload arriving during the slot and the amount served by node m, the queue is updated as

$$Q_{m,t+1} = \max \left\{ Q_{m,t} + A_{m,t} - f_m \Delta t, 0 \right\} \tag{7}$$

The nonnegative update in Eq. (7) adds newly assigned workload and subtracts the service completed during one slot. A node queue therefore accumulates historical backlog, current task assignments, and the service provided by the node. Assigning

more computation to a node in the present slot may reduce the immediate processing cost of the current task, but may also raise queue pressure in later slots. Queue conditions therefore affect the suitability of candidate nodes over subsequent slots.

3.2 Execution Model

Task completion depends on both wireless transmission and computation, as in standard mobile edge computation offloading models [41]. The present model uses distance dependent link rate, transmission delay, and computation delay. A local node requires no wireless transfer but is constrained by its local queue and computing capacity. A remote node must additionally satisfy coverage, reliable rate, and maximum transmission time requirements. The source-to-node distance used for these checks is calculated as

$$\delta_{i,m,t} = \| \mathbf{p}_{s_i,t} - \mathbf{p}_{m,t} \|_2 \tag{8}$$

Here, $\delta_{i,m,t}$ measures the Euclidean distance between the source and candidate node. Using this distance, the effective wireless rate from s_i to node m is approximated as

$$R_{i,m,t} = \mathrm{clip}\left(B_m \log_2\left[1 + \frac{P_m h_0}{\sigma^2(\delta_{i,m,t} + \delta_0)^\alpha}\right], R_{\mathrm{floor}}, R_{\max} \right) \tag{9}$$

In Eq. (9), B_m, P_m, h_0, σ^2, and α denote the effective bandwidth, node transmit power, reference channel gain, noise power, and path loss exponent, respectively. The numerical rate bounds are $R_{\mathrm{floor}} = 0.2$ Mbps and $R_{\max} = 80$ Mbps, with $\delta_0 = 1$ m and $\alpha = 2.05$ in the implementation. The reliability threshold $R_{\min} = 1.35$ Mbps is separate from the numerical clipping bound and is used only in physical feasibility assessment.

Figure 3 illustrates task partitioning and parallel local and remote execution.

When a fraction $\rho_{i,t}$ of the task is offloaded to remote node m, the transmission time can be calculated as

$$T^{tr}_{i,m,t} = \frac{\rho_{i,t} d_{i,t}}{R_{i,m,t}} \tag{10}$$

The expression in Eq. (10) converts the transmitted data volume into time using the selected link rate. With local backlog included, completion time on the source vehicle is given by

$$T^{loc}_{i,t} = \frac{Q_{0,t} + (1 - \rho_{i,t})\, C_{i,t}}{f_{0,t}} \tag{11}$$

Thus, local execution covers both the queued workload and the task fraction retained at the source.

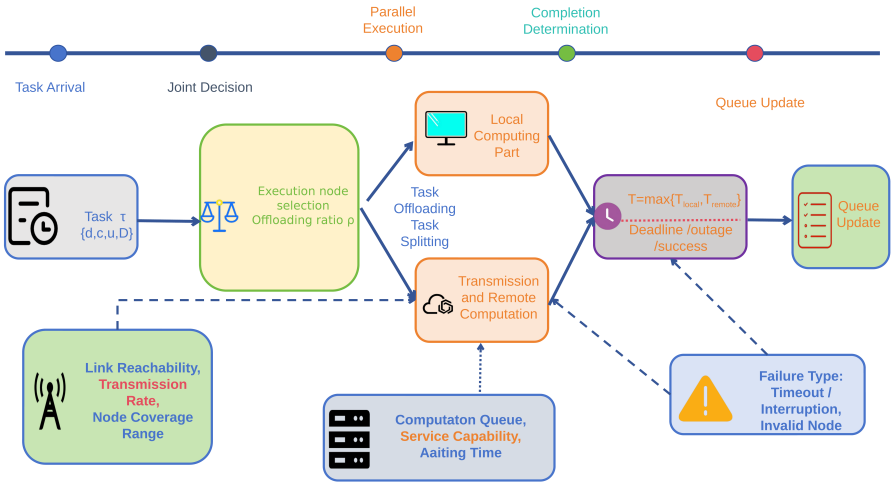


Fig. 3 Parallel execution of a partitioned task

For remote execution, transmission is followed by processing at the selected node, giving

$$T_{i,m,t}^{off} = T_{i,m,t}^{tr} + \frac{Q_{m,t} + \rho_{i,t} C_{i,t}}{f_{m,t}} \tag{12}$$

The remote branch therefore accounts for the wireless transfer and the workload waiting or executing at node m. Because both task fractions execute in parallel, the overall completion time is determined by

$$T_{i,t}^{total} = \max \left\{ T_{i,t}^{loc}, T_{i,m,t}^{off} \right\} \tag{13}$$

The maximum in Eq. (13) captures the slower of the two simultaneous branches; task completion is not achieved until both parts finish.

The split can change with task size, queue state, and the link to the selected node. A failure is recorded when the candidate is unreachable, the reliable rate condition is missed, or the deadline expires. These records separate link failures from deadline failures in the later analysis.

3.3 Decision Model

The decision state must describe the current task, candidate resources, spatial relations, queue conditions, and the task flow variables needed to represent changes over a short horizon. At this stage, the model specifies what information and control variables are available, without prescribing how a particular policy architecture should process them.

For each UAV k, the local observation $o_{k,t}$ contains the task attributes available at decision time, position and reachability information, and the current resource pool state. The joint decision consists of a choice of execution node, UAV displacement,

and an adjustment of the offloading ratio. These variables have different action forms and are coupled because the selected execution resource changes the link, queue, and resource conditions faced by the continuous control variables. Section 4 describes how PGHO organizes these generic decision variables into upper and lower policies.

The immediate reward gives priority to reliability. Delay margin and energy efficiency benefits are available only when the task succeeds. Penalties distinguish failure causes, invalid or infeasible actions, UAV movement, and the gap in estimated completion delay. This prevents the policy from obtaining an apparently low delay merely by reducing offloading or repeatedly choosing one node. Task success is consequently defined by a joint deadline and feasibility condition:

$$I_{i,t}^{succ} = \mathbf{1}\left\{T_{i,t}^{total} \le D_{i,t} \ \wedge \ I_{i,t}^{out} + I_{i,t}^{inv} + I_{i,t}^{Hinv} = 0\right\} \tag{14}$$

The indicator in Eq. (14) equals one only if the deadline is met without link interruption, invalid node selection, or an infeasible upper action. Deadline failure, link interruption, an invalid node, and an infeasible upper action are recorded with separate indicators. Additional indicators distinguish remote execution, local execution, and future high pressure, so the reward retains the source of both reliability penalties and execution benefits.

The normalized computation and communication energy cost reported in the result tables includes only normalized local computation, remote computation, and wireless transmission costs. UAV movement cost participates in policy optimization but is not added to this reported metric. The metric is therefore not interpreted as the complete energy consumption of the UAV system and is not expressed in physical joules.

Reward calculation uses normalized completion delay and normalized computation and communication energy. The reference quantity E_{ref} places computation and transmission costs on a common scale. The two normalized terms are calculated as

$$\overline{T}_{i,t} = \frac{T_{i,t}^{total}}{D_{i,t}}, \qquad \overline{E}_{i,t} = \frac{E_{i,t}^{loc} + E_{i,t}^{off} + E_{i,t}^{tr}}{E_{ref}} \tag{15}$$

The first ratio measures delay relative to the task deadline; the second measures computation and communication cost relative to E_{ref}. These normalizations allow delay and energy to enter the reward on comparable scales.

Combining the reliability and efficiency benefits with failure and action penalties, the immediate reward is written as

$$r_{i,t} = R_{i,t}^{succ} + R_{i,t}^{eff} - C_{i,t}^{fail} - C_{i,t}^{act} \tag{16}$$

In Eq. (16), the positive terms reward successful and efficient execution, whereas the negative terms penalize failures and unfavorable actions. The basic value of the success term is 12 for success and -12 for failure. The efficiency term includes delay margin and normalized energy only when the task succeeds. The failure cost combines deadline failure, link interruption, and invalid or infeasible actions. The action cost combines UAV movement, estimated delay difference, a very small offloading ratio, and future pressure. The fixed coefficients preserve reliability as the primary objective and are used consistently across the reported PGHO experiments. Evaluation across methods is based on task success ratio, average delay, 95th percentile delay (P_{95}),

99th percentile delay (P_{99}), and normalized energy cost. Policy optimization return is excluded because the learning methods use different reward constructions and Task Flow Greedy (TF Greedy) has no reinforcement learning return.

3.4 Problem Formulation

Let $\mathcal{A}_t \subseteq \mathcal{M}$ denote the candidate execution nodes available in slot t before the physical screening used by PGHO. The method then constructs a feasible subset $\mathcal{A}_t^{\text{mask}} \subseteq \mathcal{A}_t$, defined in Section 4.2, before the upper node decision.

The joint decision in slot t consists of a discrete execution node, a continuous offloading ratio, and UAV displacement, and it belongs to the feasible set $\mathcal{U}_t$. The joint control vector is expressed as

$$\mathbf{u}_t = (x_{i,t}, \rho_{i,t}, \Delta\mathbf{p}_{u,t}) \in \mathcal{U}_t \tag{17}$$

The three components in Eq. (17) correspond to node selection and continuous control. In particular, $x_{i,t}$, $\rho_{i,t}$, and $\Delta\mathbf{p}_{u,t}$ denote the selected node, offloading ratio, and UAV displacement, respectively. The set $\mathcal{U}_t$ enforces node availability, ratio limits, UAV displacement, and the permitted flight region. Coverage, reliable rate, and transmission time are checked later through $\mathcal{A}_t^{\text{mask}}$. Task flow, queues, and vehicle and UAV positions continue to evolve after the decision. An action taken in one slot therefore changes part of the state used in the next. The objective is the discounted return over the decision horizon. Delay, normalized energy, deadline failure, and task completion enter through the reward. The objective over the decision horizon can be written as

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \tag{18}$$

The expectation in Eq. (18) accumulates discounted slot rewards under policy π, thereby accounting for the influence of each action on later states.

The admissible node, ratio, displacement, and flight region conditions can be expressed together as

$$\begin{aligned} \mathcal{U}_t = \{(x, \rho, \Delta\mathbf{p}) : x \in \mathcal{A}_t,\ 0 \le \rho \le 1,\ \|\Delta\mathbf{p}\|_2 \le v_{\max}\Delta t, \\ \mathbf{p}_u + \Delta\mathbf{p} \in \Omega_u\} \end{aligned} \tag{19}$$

The constraints in Eq. (19) limit the available execution node, the offloading proportion, and the UAV movement within each slot.

The problem contains discrete node selection, a continuous offloading ratio, and UAV movement control. The objective is also affected by future task changes, link conditions, and queue evolution. The problem is therefore a nonconvex sequential decision problem with mixed discrete and continuous variables. In the present problem, the discrete choice of execution node determines part of the link, queue, and resource conditions under which the continuous variables are selected. This dependency motivates the solution structure introduced in Section 4.

4 Proposed PGHO Method

4.1 Future Task Pressure Representation

To describe the temporal correlation of the task flow, this paper records four online sequences: task computational load, urgency, arrival intensity, and deadline. Each sequence keeps a history buffer of at most 96 samples. A short horizon of 4 slots is predicted with three time scales for recent bursts, medium-term changes, and longer trends.

In PGHO, predicted workload, predicted urgency, arrival pressure, deadline pressure, and the future task flow pressure index (FTPI) are included in the observation. The predicted task information is included in the upper decision state and contributes to candidate evaluation before node selection. The predictive information also enters lower continuous control directly and affects the future risk and reward terms that use the predicted information. Prediction therefore remains active through node evaluation and action generation instead of appearing once as an independent preprocessing output.

For any observed sequence, the predictor uses three time scales. The prediction horizon is $H = 4$, the maximum history length is $L_h = 96$, the three window lengths are $(w_s, w_m, w_l) = (3, 6, 12)$, the smoothing factors are $(\alpha_s, \alpha_m, \alpha_l) = (0.70, 0.48, 0.30)$, the fusion weights are $(\omega_s, \omega_m, \omega_l) = (0.50, 0.30, 0.20)$, and the trend decay factor is $\delta = 0.82$.

Each time scale calculates an exponentially weighted moving average (EWMA) level term and a window trend term. When the history is shorter than the window, the available part is used. Trend decay limits excessive extrapolation of short-term anomalies. The results from the three scales are combined with fixed weights to balance response speed and fluctuation suppression.

For each temporal scale k, the smoothed level is obtained from the recent observations by

$$\ell_t^{(k)} = \mathrm{EWMA}_{\alpha_k}\left(y_{t-w_k+1:t}\right), \qquad k \in \{s, m, l\} \tag{20}$$

Here, α_k controls how strongly the EWMA responds to newer observations. The corresponding trend is estimated from the change across the selected window:

$$g_t^{(k)} = \frac{y_t - y_{t-w_k+1}}{\max\{1, w_k - 1\}} \tag{21}$$

The difference in Eq. (21) expresses the average change per slot within the window. To predict h slots ahead, the three levels and decayed trends are combined as follows:

$$\widehat{y}_{t+h} = \mathrm{clip}\left(\sum_{k\in\{s,m,l\}} \omega_k \left[\ell_t^{(k)} + h\delta^{h-1} g_t^{(k)}\right], y_{\min}, y_{\max}\right) \tag{22}$$

The weighted sum in Eq. (22) balances recent and longer trends, while the clip operation keeps each prediction within its valid range.

The same prediction process is applied to task computational load, urgency, arrival intensity, and deadline, giving four sequences for short-term prediction. The four variables describe future task pressure from different aspects. Computational load shows

the processing demand of one task, arrival intensity shows the pressure from the number of tasks, urgency affects resource priority, and the deadline shows the allowed completion time. The four resulting prediction sequences are collected as

$$\widehat{\mathbf{y}}_t = \left(\widehat{\mathbf{L}}_{t+1:t+H}, \widehat{\mathbf{u}}_{t+1:t+H}, \widehat{\mathbf{A}}_{t+1:t+H}, \widehat{\mathbf{D}}_{t+1:t+H}\right) \tag{23}$$

These four sequences retain the forecast for each task attribute and provide the inputs used to construct the pressure features.

From the four predicted sequences, this paper extracts cumulative load, peak load, mean arrival intensity, peak urgency, deadline pressure, and burstiness to form a compact pressure vector. After fixed scale normalization, the components are combined by weight vector $\mathbf{w}$ into FTPI. The original predicted sequences remain in the state. FTPI only gives a compact expression of risk over the short prediction horizon and does not replace the original information. The pressure components are organized into the vector

$$\mathbf{q}_t = \left(L_t^{cum}, L_t^{peak}, A_t^{mean}, u_t^{peak}, P_t^{ddl}, B_t\right) \tag{24}$$

The components have different ranges, so they are first normalized before their weighted combination:

$$\widetilde{\mathbf{q}}_t = \mathrm{Norm}\left(\mathbf{q}_t\right) \tag{25}$$

The normalized pressure terms are then aggregated into a bounded index:

$$\mathrm{FTPI}_t = \mathrm{clip}\left(\mathbf{w}^{\mathsf{T}}\widetilde{\mathbf{q}}_t, 0, 2\right) \tag{26}$$

The resulting FTPI in Eq. (26) is a compact summary of the six pressure components, not a substitute for the four predicted sequences. The deadline is treated as an observed online sequence and is processed by the same predictor rather than being inferred from urgency alone. The task-flow generation rules used to create the experimental pressure scenarios are given in Section 5.1.

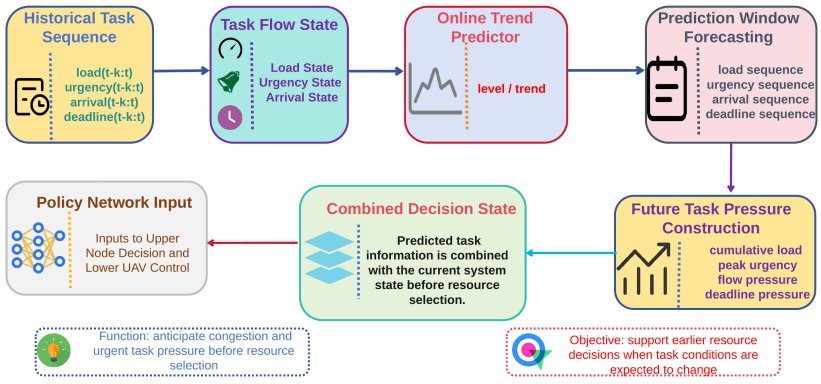


Fig. 4 Future task pressure representation

Urgency remains a continuous state variable. For thresholded reward handling and auxiliary success statistics, tasks with $u_{i,t} \geq 0.65$ are treated as high urgency tasks. This thresholded category is not reported as a separate primary metric. Figure 4 shows the future task pressure representation. The four predicted task sequences are summarized into pressure features so that workload, task arrivals, urgency, and deadline changes can be considered together before resource selection. The resulting information enters state construction, risk evaluation, and both policy levels.

4.2 Hierarchical Decision Making with Predicted Task Pressure

The upper decision is made first. Its state contains current heterogeneous resource conditions together with the future task pressure representation, and the selected resource is restricted by the dynamic feasible set. The lower controller is then formed around that choice: the relevant link, queue, resource availability, and ratio bounds enter its observation, while future task pressure remains available. UAV displacement and the offloading ratio are generated under the conditions associated with the selected node. The hierarchy therefore follows the dependency between resource selection and continuous control rather than simply assigning different action types to different learning algorithms.

The upper policy produces the discrete decision on the execution node,

$$a_{i,t}^{H} = \pi_H\left(\mathbf{s}_t\right), \qquad a_{i,t}^{H} \in \mathcal{A}_t^{\text{mask}}, \tag{27}$$

and the lower policy uses the conditions of the selected node together with the local observation to generate UAV displacement and the adjustment of the offloading ratio:

$$\mathbf{a}_{k,t}^{L} = \pi_k^{L}\left(\mathbf{o}_{k,t}, a_{i,t}^{H}\right), \qquad \mathbf{a}_{k,t}^{L} = \left(\Delta x_{k,t}, \Delta y_{k,t}, \eta_{k,t}\right). \tag{28}$$

Equations (27) and (28) implement the dependency identified in Section 3.4: the execution resource is selected first, and the continuous action is then generated under the conditions associated with that choice. During learning, both policy levels use the slot reward defined in Eq. (16).

The offloading ratio is zero for local execution. For remote execution, the environment first selects a baseline from a fixed set of candidate ratios. The mean ratio residual produced by the UAVs is multiplied by 0.16 and added to that baseline, then clipped by dynamic bounds determined from task pressure, link quality, and the node queue. The lower policy makes a continuous correction around a feasible baseline rather than searching the entire ratio interval. Ratio averaging and dynamic bounds incorporate task pressure, link quality, and queue conditions directly into the continuous offloading decision.

The upper and lower policies are trained with the standard DDQN and MADDPG update rules used by the hierarchical controller. The upper DDQN first selects the next feasible action using its online Q network and evaluates that action with its target network. The resulting temporal difference target is calculated as follows:

$$y_t^H = r_t + \gamma(1-d_t)Q_{\theta^-}^H\left(\mathbf{s}_{t+1}, \arg\max_{a\in\mathcal{A}_{t+1}^{\text{mask}}} Q_\theta^H(\mathbf{s}_{t+1}, a)\right) \tag{29}$$

Here, d_t is the episode termination indicator, and θ and θ^- identify the parameters of the online and target upper Q networks, respectively. The upper network is trained by minimizing the corresponding Smooth-L1 loss:

$$\mathcal{L}_H(\theta) = \mathbb{E}\left[\text{SmoothL1}\left(y_t^H - Q_\theta^H\left(\mathbf{s}_t, a_t^H\right)\right)\right] \tag{30}$$

The loss in Eq. (30) penalizes the difference between the estimated upper Q value and its target. For the continuous lower control, the target value of UAV k is defined by

$$y_{k,t}^L = r_t + \gamma\left(1-d_t\right) Q_{k,\phi_k^-}\left(\mathbf{s}_{t+1}, \mu^-\left(\mathbf{o}_{t+1}\right)\right) \tag{31}$$

Here, ϕ_k^- denotes the target Critic parameters, and μ^- denotes the target Actor output. The online Critic, parameterized by ϕ_k, is updated using the loss

$$\mathcal{L}_{Q_k}\left(\phi_k\right) = \mathbb{E}\left[\text{SmoothL1}\left(Q_{k,\phi_k}\left(\mathbf{s}_t, \mathbf{a}_t\right) - y_{k,t}^L\right)\right] \tag{32}$$

The loss in Eq. (32) fits the Critic to the target for UAV k. The corresponding Actor parameters ψ_k are updated through the deterministic policy gradient:

$$\nabla_{\psi_k} J_k = \mathbb{E}[\nabla_{a_k} Q_{k,\phi_k}(\mathbf{s}_t, \mathbf{a}_t)\, \nabla_{\psi_k}\mu_{\psi_k}(\mathbf{o}_{k,t})] \tag{33}$$

The gradient in Eq. (33) adjusts the Actor through the sensitivity of the centralized Critic to the UAV action. Both value losses use Smooth-L1 to limit the influence of occasional large temporal difference errors during training.

The upper level uses an online Q network to select the next node and a target Q network to evaluate the target value. The lower Critic evaluates the joint action of all UAVs, while each Actor updates continuous control from its local observation. Target networks are updated through soft parameter synchronization, which stabilizes the value and policy updates during training.

Figure 5 emphasizes decision conditions rather than network architecture. Future task pressure contributes to upper node evaluation and remains available to lower continuous control. The dynamic feasible set is formed from physical feasibility and estimated execution conditions before the upper policy ranks the remaining candidates. The node chosen by the upper policy constrains the lower action through link state, queue state, resource availability, and the feasible ratio range. DDQN and MADDPG implement the discrete and continuous policies, respectively, within this decision sequence.

The upper Q network has two hidden layers with 256 units each. Each lower Actor has two hidden layers with 192 units, and the Critic has two hidden layers with 256 units. DDQN and MADDPG both use replay buffers and target networks, whose parameters are synchronized gradually by soft updates. Future task pressure enters both upper selection and lower control, while the upper action specifies the execution conditions under which the continuous decision is generated.

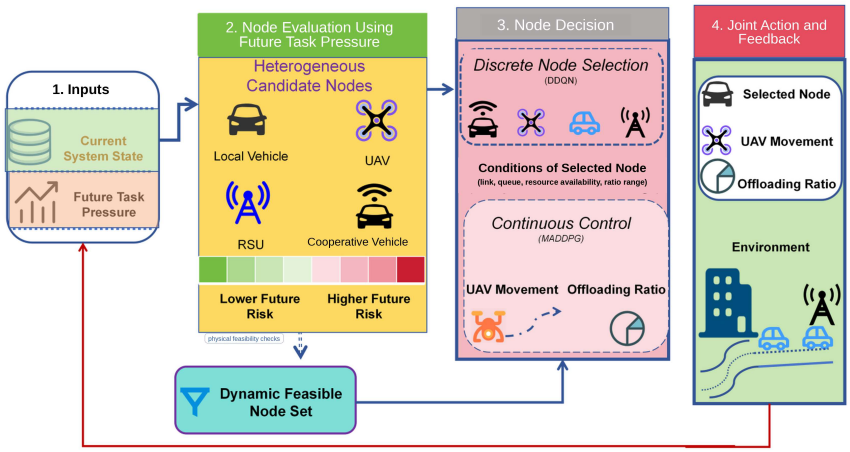


Fig. 5 Hierarchical offloading using future task pressure and the conditions of the selected node

In a dynamic vehicular network, not every candidate node is available in every slot. PGHO first identifies nodes satisfying coverage distance, minimum reliable rate, and maximum reliable transmission time. The physical feasible set is defined by

$$\mathcal{A}_t^{\text{mask}} = \{a \in \mathcal{A}_t : \delta_{a,t} \le R_a^{\text{cov}},\ R_{a,t} \ge R_{\min},\ T_{a,t}^{\text{tr}} \le \tau_{\max}^{\text{tr}}\} \cup \{0\} \tag{34}$$

Here, R_a^{cov} denotes the coverage range of candidate a, and index zero corresponds to local execution. The set in Eq. (34) represents physical feasibility only. PGHO then uses estimated completion delay to screen preferred candidates. If no candidate satisfies the estimated time limit, a small number of nodes with lower estimated delay are kept from the physical feasible set as the fallback set. The subsequent screen based on estimated delay and the fallback rule further restrict this physically feasible set before node selection.

Within the feasible candidate set, the upper value uses a bounded processing cost correction to reduce the probability of selecting a node with clearly high delay:

$$a_t^H = \arg\max_{a \in \mathcal{A}_t^{mask}} \left[Q_\theta^H(\mathbf{s}_t, a) - \lambda_{prior} c_t^{prior}(a) \right] \tag{35}$$

The selection rule in Eq. (35) uses the learned Q value while applying a bounded penalty to candidates with high estimated processing cost. The correction may change the ordering of candidates; it does not alter the physical feasibility screening.

4.3 Policy Selection Using Validation Results

Policy selection uses periodic validation rather than simply retaining the policy obtained at the end of training. During training, the current policy is evaluated on a fixed validation flow, and an improvement creates a new checkpoint. A sequence of evaluations without improvement, together with a persistent decline, can restore a

checkpoint with better validation performance. The validation procedure ends before deployment and does not alter the online hierarchical decision. Figure 6 shows how periodic validation, checkpoint retention, degradation detection, and restoration are used for final policy selection. The diagram separates policy selection during training from the online hierarchical decision used after training.

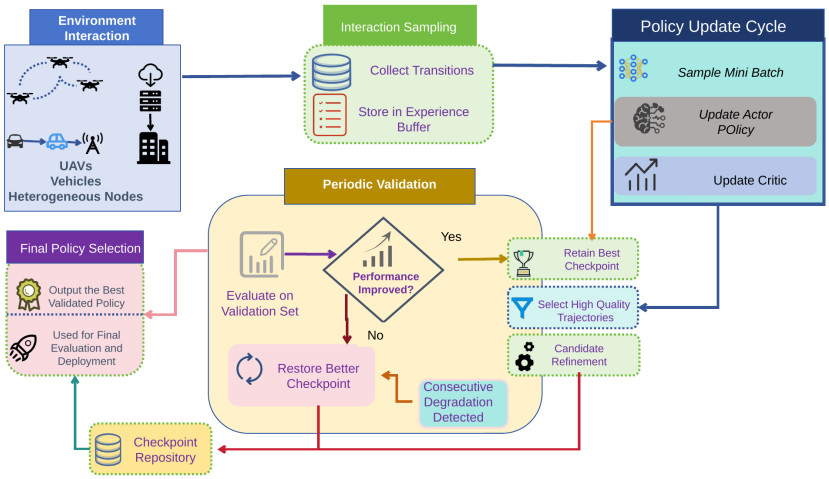


Fig. 6 Policy selection using checkpoint evaluation

Periodic validation uses a score that prioritizes reliability while also accounting for average and tail delays. The checkpoint retention score can be expressed as follows: C_{rel} combines the success ratio threshold and penalties for deadline failure, link interruption, and invalid upper actions. Δ_{bal} combines the balance terms for offloading ratio and node use. The fixed weights follow the checkpoint retention score used during the main training stage, while reliability conditions are evaluated jointly with the aggregate score.

$$S(\pi) = 165R_{succ} - C_{rel} - 20T_{avg} - 5T_{P95} - 4E_{avg} + \Delta_{bal} \tag{36}$$

In Eq. (36), T_{P95} is the 95th percentile delay. A higher score reflects the selected balance of reliability, delay, energy, and policy use. Besides the aggregate score, candidate policies are also retained according to reliability, average delay, the 95th percentile delay P_{95}, the 99th percentile delay P_{99}, normalized energy, and policy return. Candidate policies are first compared by task success, then by deadline failure, link interruption, and invalid upper actions, and finally by delay, energy, offloading ratio, and node use.

Checkpoint selection within PGHO uses the validation score in Eq. (36). Evaluation across methods instead uses the common physical metrics reported in the experiments, because policy returns and validation scores are not directly comparable across methods with different reward constructions and TF Greedy has no reinforcement learning return.

Periodic evaluation produces a candidate set using different criteria such as reliability, aggregate score, average delay, and tail delay. Using several criteria prevents the candidate set from concentrating around policies selected by a single metric.

Each retained policy is then continued under predefined refinement settings and evaluated again on the same validation flow. During refinement, candidate ranking keeps reliability as the primary requirement but places stronger weight on average delay, P95 delay, and P99 delay than the checkpoint retention score in Eq. (36). The selected checkpoint is then used for the final evaluation. The variant without selection inheritance (PGHO no SI) removes checkpoint retention, restoration, and candidate refinement together. The result of PGHO no SI therefore evaluates the complete procedure and cannot identify rollback, inherited parameters, or one refinement setting as the sole cause.

Algorithm 1 summarizes the execution order of the procedure based on validation results, including checkpoint update, restoration after persistent degradation, candidate refinement, and final policy selection.

Algorithm 1 Policy selection for PGHO using validation results

Initialize the predictor, policy networks, replay storage, validation log, and checkpoint pool.
repeat
    Build the current state and the future pressure features used to assess heterogeneous nodes.
    Build the dynamic feasible node set through physical feasibility checks, comparison using predicted task information, and action masking.
    Select the execution node with the upper policy, then generate UAV movement and offloading ratios under the selected node conditions.
    Execute the joint action, store transitions, and update the upper and lower policy networks.
    Evaluate the current candidate periodically on the fixed validation task flow.
    Save the checkpoint when the validation score improves.
    Restore the historical best checkpoint when the persistent degradation criterion is met.
until the main training criterion is met
Construct a diverse candidate pool from retained checkpoints using different evaluation criteria.
for each candidate strategy in the pool **do**
    Refine the candidate using predefined refinement settings.
    Evaluate the refined candidate on the same fixed validation task flow.
    Update the global best checkpoint when the reliability and delay ranking during refinement improves.
end for
Restore the global best checkpoint as the selected PGHO policy.

The checkpoint comparison and candidate refinement stages of Algorithm 1 are completed before deployment, so they do not add an online control branch to the deployed policy.

The online complexity of the final policy consists of prediction updates, candidate execution estimates, the upper node decision, and forward evaluation of lower continuous control. Let the unit costs of prediction update, one candidate execution estimate, one upper forward pass, and one lower Actor forward pass be C_P, C_E, C_H and C_L.

For N candidate nodes, G fixed ratio evaluation points per node, and N_u UAVs, the decision complexity per slot is $O(C_P + N \cdot G \cdot C_E + C_H + N_u \cdot C_L)$. Policy selection is completed before the final policy is selected and does not change the parameter size of the deployed networks. Online computation is dominated by candidate execution estimation over fixed size sets.

5 Results and Discussion

Section 5 first compares PGHO with TF Greedy and learning baselines under identical task flows and a common evaluation protocol. It then analyzes the main component variants and finally examines the evaluation traces over episodes. The variant without prediction enhancement (PGHO no PE) removes predicted task information from the decision process. The variant without hierarchical decision making (PGHO no HD) removes the hierarchical decision structure. The variant without action masking (PGHO no AM) removes action masking. PGHO no SI removes the validation selection procedure. Elite experience and policy rollback remain auxiliary operations within the training procedure.

5.1 Experimental Setup

The experiments use normal load, high load, and tight deadline task pressure scenarios with random seeds 42, 43, and 44. For each generated task, the initial deadline depends on urgency, load state, and burst state. Before scenario adjustments, it is calculated as

$$D_{i,t}^{base} = \mathrm{clip}\left(6.50 - 2.40u_{i,t} - 0.65l_t - 0.25b_t + \varepsilon_{i,t}^{D},\ 2.50,\ 6.50\right). \tag{37}$$

The clipped expression in Eq. (37) limits the baseline deadline to the specified range. The load and burst states represent the current task flow and its changes, while the random term allows tasks under the same conditions to have different deadlines. The normal scenario leaves the generated task flow unchanged. The high load scenario increases task input size and computational density by factors of 1.12 and 1.08, shifts urgency, arrival intensity, and burst state by 0.02, 0.06, and 0.08, scales the deadline by 0.96, and increases background load by a factor of 1.06. The tight deadline scenario uses factors of 1.03 and 1.02 for task size and computational density, shifts urgency, arrival intensity, and burst state by 0.08, 0.02, and 0.03, scales the deadline by 0.88, and increases background load by a factor of 1.02. After scenario adjustment, deadlines are clipped to the interval from 2.20 s to 6.50 s. The predictor processes the resulting deadline sequence online. The main comparison methods are TF Greedy, MADDPG, soft actor critic (SAC), hybrid multiagent proximal policy optimization (H MAPPO), and PGHO. SAC uses maximum entropy continuous control [42], while the H MAPPO baseline is adapted from multiagent proximal policy optimization (MAPPO) [43]. Recent vehicular edge research has investigated greedy computation offloading [44] and applied MADDPG to joint offloading and resource allocation [45]. TF Greedy performs scheduling using fixed rules from current resource states and task estimates over a short horizon, whereas MADDPG encodes UAV movement, node preference, and offloading ratio in one continuous action. Recent UAV MEC research has applied SAC to computation offloading and flight related control [46]. Recent multi-UAV MEC research has applied MAPPO to collaborative trajectory scheduling [47]. In this implementation, H MAPPO selects a node with a discrete policy and produces UAV movement and the offloading ratio with a continuous policy. PGHO differs by more than its network choice. PGHO compares heterogeneous resources through future task pressure before node selection and passes the conditions created by that choice to the lower controller.

All component variants use the same environment and evaluation protocol. PGHO no PE removes predicted task information from the decision process. Terms that require predicted information are therefore inactive in this variant. PGHO no HD keeps the predicted task information, action masking, validation settings, task flows, and random seeds, but removes the separate upper DDQN and the lower controller that operates under the conditions of the selected node. One flat MADDPG policy produces UAV movement, node preference, and the offloading ratio. This variant is used only to measure the effect of the hierarchical decision structure. PGHO no SI removes periodic evaluation inheritance, policy recovery, and candidate refinement. PGHO no AM removes upper action masking. The variants without elite experience assistance (PGHO no EI) and without rollback (PGHO no RB) remove these two auxiliary operations, respectively. All other settings follow the corresponding PGHO experiment unless the removed mechanism requires a structural change.

To reduce variation caused by different task flows, all methods share paired task flows and background load sequences under the same scenario and seed. After training, every scenario is evaluated for 1000 episodes on a reserved fixed task flow with no parameter updates. These 1000 episodes belong to one continuous evaluation trace and should not be treated as 1000 independent random samples. PGHO selects its final policy from candidate checkpoints through the validation procedure described earlier in Section 4, whereas PGHO no SI uses the policy obtained from a single training path.

The evaluation metrics are average delay, P95 delay, P99 delay, task success ratio, and normalized energy cost. The normalized energy cost includes normalized local computation, remote computation, and wireless transmission costs only. Each scenario reports the mean and sample standard deviation over three seeds. The arithmetic mean across the three scenarios is reported without an overall standard deviation, and policy returns are not compared because they are not commensurate across methods.

Table 1 summarizes the physical parameters of tasks, links, and heterogeneous nodes. The same physical environment, communication coverage, time scale, and feasibility limits are used for all compared methods.

Table 1 defines the shared physical basis of the comparison. Its task, node, coverage, and feasibility parameters remain unchanged across the scheduling methods, while method-specific learning settings are listed separately in Table 2.

Table 2 gives the network architecture and optimization settings used by the learning methods. The table separates policy structure, replay capacity, learning rates, exploration, and prediction settings from the common physical system parameters in Table 1.

Table 1 System parameters

Symbol	Description	Value
N_s	Number of task source vehicles	10
N_u	Number of UAV edge nodes	2
N_c	Number of cooperative vehicle nodes	2
N_r	Number of RSU nodes	2
N	Number of candidate execution nodes	7
f_{local}	Computing capacity of the local vehicle	7 GHz
f_{uav}	Computing capacity of each UAV node	25 GHz
f_{cv}	Computing capacity of each cooperative vehicle	15 GHz
f_{rsu}	Computing capacity of each RSU node	40 GHz
$R_{\text{cov,UAV}}$	Communication range of a UAV node	500 m
$R_{\text{cov,CV}}$	Communication range of a cooperative vehicle	320 m
$R_{\text{cov,RSU}}$	Communication range of an RSU node	420 m
T	Time slots per episode	30
L	Evaluation episodes	1000
Scenarios	Task pressure scenarios	normal, high load, tight deadline
Seeds	Random seeds for repeated evaluation	42, 43, 44
Δt	Duration of one time slot	1 s
$v_u^{\max}$	Maximum horizontal movement of each UAV per slot	25 m/slot
$R_{\min}$	Minimum reliable transmission rate used in feasibility assessment	1.35 Mbps
$\tau_{\text{tx}}^{\max}$	Maximum reliable transmission time used in feasibility assessment	1.85 s

Table 2 Network and policy optimization configurations

Configuration Item	Setting	Description
Upper DDQN architecture	256×256, rectified linear unit (ReLU), output dimension N	Upper discrete node selection in PGHO
Lower Actor architecture	192×192, ReLU, hyperbolic tangent output	One independent Actor for each UAV
Lower Critic architecture	256×256, ReLU, centralized input	Centralized value estimation in MADDPG
Replay buffer capacity	PGHO 60,000; MADDPG 60,000; SAC 80,000	Maintained separately for each method
Discount factor and soft update coefficient	$\gamma = 0.97$, $\tau = 0.01$	DDQN, MADDPG, and SAC
Batch size	PGHO upper and lower policies 64; MADDPG and SAC 128; H MAPPO 128	Main policy update parameters
PGHO learning rates	DDQN 7×10^{-4}; Actor 7×10^{-4}; Critic 1.0×10^{-3}	DDQN, Actor, and Critic
MADDPG and SAC learning rates	MADDPG Actor 8×10^{-4}, Critic 1.2×10^{-3}; SAC Actor 5×10^{-4}, Critic 1.0×10^{-3}, temperature 3×10^{-4}	Main parameters of the comparison methods
H MAPPO network parameters	Discrete branch 256×256; continuous branch 192×192; Actor 3×10^{-4}; Critic 6×10^{-4}	Mixed-action comparison method
Upper ε exploration	$0.30 \rightarrow 0.02$, linearly decayed over the first 65 percent of episodes	Hierarchical PGHO variant
Lower continuous noise	$0.12 \rightarrow 0.015$, linearly decayed over the first 65 percent of episodes	Hierarchical PGHO variant
Prediction windows and fusion weights	Horizon $H = 4$; windows 3, 6, 12; weights 0.50, 0.30, 0.20; trend decay 0.82	Online prediction with multiple time scales

Table 2 separates the learning process from the common physical environment in Table 1. These optimization settings determine how each learning method is trained, whereas the final comparison uses the same physical performance metrics for all methods.

Table 3 summarizes the structural differences among the baselines, the complete PGHO method, and the component variants. TF Greedy uses scheduling rules based on current resource states and task estimates over a short horizon. MADDPG is a flat multi-agent continuous baseline that jointly outputs UAV movement, node preference, and offloading ratio. SAC is a maximum entropy continuous control baseline. H MAPPO is an adaptation with discrete node selection and continuous movement and offloading control. PGHO uses predicted task information during resource selection, hierarchical offloading under the conditions of the selected node, and policy selection based on validation results. PGHO no HD removes the separate upper and lower decision structure and uses one flat policy while retaining the predicted task information. PGHO no PE removes predicted task information, and terms that require these predictions are inactive accordingly. PGHO no SI removes periodic validation inheritance, policy recovery, and candidate refinement. PGHO no AM removes upper physical feasibility masking.

Table 3 Main method configurations

Method	Forecast	Node condition	Validation	Mask
TF Greedy	Rules	No	Not applicable	Not applicable
MADDPG	No	No	No	No
SAC	No	No	No	No
H MAPPO	No	Yes	No	No
PGHO	Yes	Yes	Yes	Yes
PGHO no HD	Yes	No	Yes	Yes
PGHO no PE	No	Yes	Yes	Yes
PGHO no SI	Yes	Yes	No	Yes
PGHO no AM	Yes	Yes	Yes	No

The auxiliary variants PGHO no EI and PGHO no RB retain all four main mechanisms shown in Table 3. They differ from PGHO only by removing elite experience assistance or policy rollback and are therefore discussed separately in the component results.

5.2 Comparison with Baseline Methods

The three task pressure scenarios are reported separately so that the effect of workload and deadline pressure can be distinguished from variation caused by random initialization. Each scenario gives the mean and sample standard deviation over three random seeds. The arithmetic mean of the three scenario means is used only as an overall summary and does not replace the individual scenario results. This subsection compares PGHO with TF Greedy and the learning baselines before the component analysis and evaluation traces are examined separately.

Table 4 Normal scenario results, mean ± sample standard deviation

Method	Average Delay (s)	P95 Delay (s)	P99 Delay (s)	Norm. energy cost	Task Success Ratio (%)
TF Greedy	0.409±0.000	0.636±0.003	0.750±0.004	0.709±0.001	97.088±0.100
MADDPG	0.510±0.066	0.853±0.157	0.987±0.194	0.973±0.142	88.002±17.035
SAC	0.465±0.028	0.755±0.050	0.852±0.053	0.895±0.039	98.021±1.656
H MAPPO	0.421±0.028	0.727±0.063	0.845±0.079	0.910±0.027	98.932±0.962
PGHO	0.398±0.002	0.677±0.011	0.791±0.015	0.900±0.003	99.732±0.192
PGHO no HD	0.544±0.014	1.036±0.035	1.144±0.037	0.918±0.008	99.766±0.205
PGHO no PE	0.570±0.031	1.029±0.011	1.134±0.012	0.939±0.029	99.331±0.347
PGHO no SI	0.563±0.012	1.034±0.011	1.141±0.017	0.926±0.007	98.526±1.298
PGHO no AM	0.484±0.162	0.851±0.241	0.970±0.235	0.833±0.039	98.847±1.684
PGHO no EI	0.392±0.011	0.680±0.018	0.797±0.022	0.847±0.027	99.684±0.188
PGHO no RB	0.397±0.007	0.673±0.007	0.786±0.016	0.871±0.033	99.741±0.352

Table 4 presents the results under normal load. PGHO achieves an average delay of 0.398±0.002 s together with a task success ratio of 99.732%±0.192%. In this scenario, the available computing resources leave enough spare capacity for several methods to complete most tasks, so the difference is reflected more clearly in execution delay. PGHO selects the execution node before adjusting UAV movement and the offloading ratio. The hierarchical separation reduces the coupling among decisions and lets the continuous controller operate with the conditions of the selected resource already known. The predicted task information also helps distinguish candidates that look similar in the current slot but are likely to face different queue pressure shortly afterward. H MAPPO stays relatively close because it already separates discrete node selection from continuous control. TF Greedy is also competitive when queues are short because current link and resource states are often sufficient for a good immediate choice. Even under this favorable condition, PGHO maintains both lower mean delay and a higher task success ratio, showing that its advantage is not limited to congested operation.

Table 5 High load scenario results, mean ± sample standard deviation

Method	Average Delay (s)	P95 Delay (s)	P99 Delay (s)	Norm. energy cost	Task Success Ratio (%)
TF Greedy	0.481±0.000	0.772±0.006	0.911±0.006	0.825±0.001	97.088±0.100
MADDPG	0.556±0.005	0.834±0.013	0.929±0.014	1.032±0.018	96.992±0.480
SAC	0.535±0.012	0.863±0.009	0.965±0.011	1.048±0.031	98.587±0.263
H MAPPO	0.524±0.047	0.964±0.142	1.110±0.153	1.078±0.028	99.519±0.213
PGHO	0.483±0.017	0.854±0.020	0.989±0.024	1.016±0.042	99.533±0.413
PGHO no HD	0.750±0.021	1.531±0.049	1.682±0.056	1.084±0.020	99.980±0.023
PGHO no PE	0.760±0.051	1.499±0.022	1.645±0.027	1.075±0.023	99.667±0.009
PGHO no SI	0.790±0.030	1.531±0.020	1.680±0.021	1.096±0.027	99.324±0.852
PGHO no AM	0.542±0.087	1.090±0.256	1.261±0.286	0.966±0.045	97.179±2.275
PGHO no EI	0.480±0.013	0.847±0.014	0.979±0.018	1.048±0.043	99.506±0.320
PGHO no RB	0.466±0.011	0.825±0.013	0.959±0.024	0.998±0.039	99.851±0.213

Table 5 reports the results under high load, where stronger resource competition and faster queue buildup make the quality of the execution node decision more important. TF Greedy gives an average delay only about 0.002 s lower than PGHO, while MADDPG gives slightly smaller P95 and P99 values. These isolated advantages are accompanied by clearly lower task completion reliability. PGHO reaches a task success ratio of 99.533%, higher than those of TF Greedy and MADDPG and essentially the same as the 99.519% of H MAPPO. Under high load, the mean delay therefore remains close to the minimum while task completion stays near 100%. When candidate nodes become congested, a poor resource choice is difficult to recover from through later UAV movement or ratio adjustment. PGHO reduces this risk by considering predicted task pressure and current feasibility before continuous control is applied. The higher normalized energy cost reflects more active use of the available air and ground resources, but that additional resource use is accompanied by higher execution reliability. The high-load results show that average delay remains low without sacrificing task completion as resource competition increases.

Table 6 Tight deadline scenario results, mean ± sample standard deviation

Method	Average Delay (s)	P95 Delay (s)	P99 Delay (s)	Norm. energy cost	Task Success Ratio (%)
TF Greedy	0.425±0.000	0.691±0.004	0.819±0.005	0.732±0.001	97.088±0.100
MADDPG	0.570±0.092	0.956±0.167	1.064±0.162	0.943±0.012	96.471±1.780
SAC	0.468±0.005	0.768±0.009	0.877±0.014	0.902±0.013	98.542±0.384
H MAPPO	0.448±0.027	0.787±0.058	0.916±0.064	0.958±0.032	99.380±0.157
PGHO	0.409±0.003	0.710±0.009	0.830±0.011	0.897±0.025	99.850±0.159
PGHO no HD	0.613±0.015	1.136±0.037	1.250±0.042	0.960±0.009	99.657±0.337
PGHO no PE	0.613±0.024	1.118±0.008	1.230±0.011	0.988±0.037	98.796±0.487
PGHO no SI	0.545±0.097	0.984±0.162	1.099±0.150	0.945±0.019	98.004±1.317
PGHO no AM	0.462±0.065	0.898±0.174	1.035±0.186	0.862±0.022	93.413±5.818
PGHO no EI	0.412±0.013	0.713±0.028	0.832±0.031	0.907±0.024	99.647±0.406
PGHO no RB	0.413±0.009	0.702±0.005	0.818±0.013	0.912±0.041	99.877±0.147

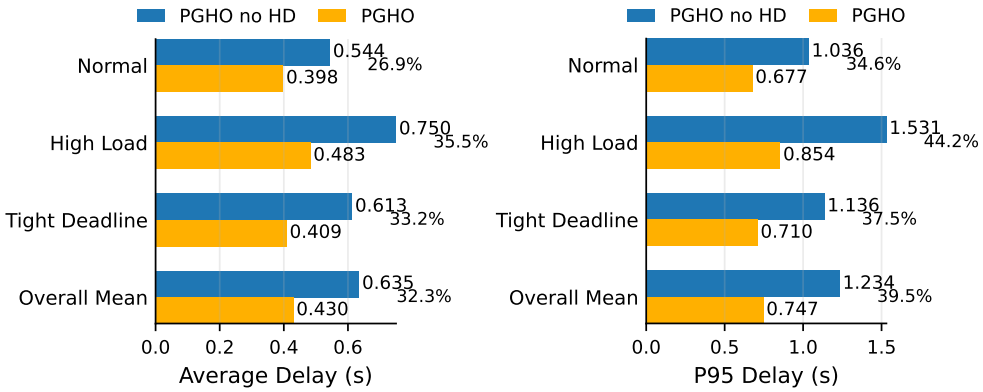
Table 6 gives the results when the task deadlines are tightened. PGHO obtains an average delay of 0.409±0.003 s and a task success ratio of 99.850%±0.159%. Its average delay is lower than those of MADDPG, SAC, and H MAPPO, while the success ratio is the highest among the principal baselines. Tight deadlines leave little time to recover after an unsuitable resource has already been selected and transmission or queueing has begun. The predicted workload, arrivals, urgency, and deadline information become useful before the task is assigned, and the action mask removes candidates whose current communication conditions already make reliable execution difficult. Future task information and action masking jointly reduce the chance that the continuous controller starts from a resource choice with little margin for adjustment. The success loss of PGHO no AM under the same scenario further supports the importance of early feasibility screening when deadline slack becomes small.

Averaged over the three scenarios, PGHO achieves 0.4301 s average delay and 99.705% task success ratio. Relative to MADDPG, SAC, and H MAPPO, average delay is reduced by 21.1%, 12.1%, and 7.4%, respectively, while task success ratio increases by 5.88, 1.32, and 0.43 percentage points. H MAPPO remains the closest

learning baseline because it already separates node selection from continuous control. TF Greedy also gives low mean delay because current queue and link states support effective immediate choices. Against TF Greedy, PGHO keeps a 1.83% advantage in average delay and improves task success ratio by about 2.62 percentage points. Taken together, the results show a trade-off between mean delay and task completion across changing workload and deadline pressure rather than the minimum value of every individual metric. The approximately 24.2% higher normalized energy cost relative to TF Greedy reflects the additional resource use associated with this reliability. The following subsection examines the main components of PGHO.

5.3 Analysis of Main Components

Fig. 7(a) isolates the effect of the hierarchical decision structure. PGHO no HD keeps the same future task information and the main evaluation settings, but removes the separate execution node decision and uses one flat policy. Introducing the hierarchy reduces average delay by 26.9%, 35.5%, and 33.2% under normal load, high load, and tight deadlines, respectively. The mean of the three scenario values decreases from 0.6355 s to 0.4301 s, a reduction of 32.3%. The size and consistency of these reductions indicate that the hierarchical separation contributes to latency reduction. Once the execution node is fixed, the lower policy only needs to adjust UAV movement and the offloading ratio for that resource condition. The flat policy must coordinate node preference and continuous control at the same time, which leaves more weak combinations to be explored. The latency benefit becomes most visible under high load, where an unsuitable resource choice leaves less room for later adjustment.



(a) (b)

Fig. 7 Impact of the hierarchical decision structure on (a) average delay and (b) P_{95} delay.

Fig. 7(b) shows an even stronger effect on tail delay. P95 delay is reduced by 34.6%, 44.2%, and 37.5% in the three scenarios, and its mean decreases from 1.2343 s to 0.7472 s, corresponding to a 39.5% reduction. The largest change again appears under high load. The larger reduction under high load indicates that separating resource

selection from continuous control becomes particularly valuable as queue imbalance and resource competition increase. PGHO no HD still maintains a very high task success ratio, so the evidence does not suggest that the hierarchy is necessary for basic task feasibility. Its main benefit in these experiments is a substantial improvement in both average and tail delay.

The validation procedure mainly affects which trained policy is retained for final evaluation. Compared with PGHO no SI, the complete method reduces the average delay across the three scenarios by 32.0% and P_{95} delay by 36.8%, while task success ratio increases by 1.09 percentage points. Under high load alone, average delay falls from 0.7900 s to 0.4833 s. These differences suggest that relying on the last training state can discard a policy with better validation performance that appeared earlier on the same task flow. Because PGHO no SI removes periodic validation, recovery, and candidate refinement together, the experiment supports the complete selection process rather than any one of those operations in isolation.

Action masking removes candidates whose current coverage, reliable rate, or transmission time already makes them poor execution choices. Compared with PGHO no AM, the complete method reduces the average delay across the three scenarios by 13.3% and P_{95} delay by 21.1%, while task success ratio increases by 3.23 percentage points. The effect is most visible under tight deadlines, where task success ratio rises from $93.413\% \pm 5.818\%$ without masking to $99.850\% \pm 0.159\%$ with the complete method. Failure statistics show that the lost tasks are mainly associated with link interruption rather than deadline violation. The main role of the mask is to prevent obviously unreliable assignments before the upper policy ranks the remaining resources. The observed reliability gain is consistent with the role of early feasibility screening when deadline slack is limited.

The additional results for elite experience and policy rollback show a more limited effect. The average delay values across the three scenarios for PGHO no EI and PGHO no RB are 0.4281 s and 0.4257 s, close to the 0.4301 s of PGHO and lower in some scenarios. Their task success ratios are 99.612% and 99.823%, respectively. The two auxiliary operations do not show a consistent independent improvement in average performance. They are therefore treated only as auxiliary components for candidate update and policy selection.

Removing the future task information increases both average and tail delay. Compared with PGHO no PE, the complete method reduces the average delay across the three scenarios by 33.6% and P_{95} delay by 38.5%, while task success ratio changes by only 0.43 percentage points. Most tasks can still be assigned to a feasible resource without prediction, but they are more often sent to nodes whose queues become less favorable in the following slots. The much larger change in average and tail delay than in completion rate is consistent with the intended role of future task pressure: it mainly improves the timing and destination of resource selection rather than basic execution feasibility. PGHO no PE therefore evaluates the overall contribution of predicted task information to the decision process and does not isolate the effect of one prediction variable or of the pressure representation alone.

5.4 Evaluation Traces and Focused Comparisons

Figure 8 shows task success ratio over 1000 evaluation episodes under high load. PGHO remains close to 99.5% throughout the trace and stays in the same reliability range as H MAPPO, while SAC, TF Greedy, and MADDPG remain lower. The curve also complements the tabled mean values by showing that the high success ratio of PGHO is sustained over the evaluation trace rather than being produced by a small number of favorable episodes. The curve is averaged over seeds 42, 43, and 44 and then smoothed with a 60 episode moving average.

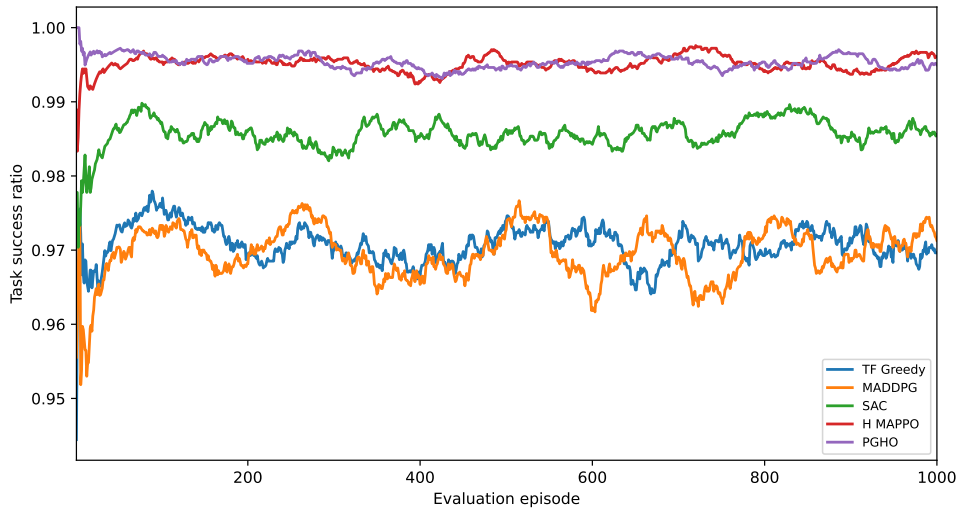


Fig. 8 Task success ratio over episodes in the high load scenario

Each scenario uses only three random seeds. The pressure scenarios for the same seed are generated from the same basic task flow by scenario rules. Because the pressure scenarios for a given seed share the same underlying task flow, the analysis is based on scenario means, sample standard deviations, the arithmetic mean of the three scenario means, and the observed metric changes rather than significance testing across the the nine combinations of scenarios and seeds.

Figure 9 compares PGHO with TF Greedy, the baseline with the closest average delay. PGHO reduces average delay by 2.62% under normal load and 3.72% under tight deadlines. Under high load, TF Greedy is lower by only 0.51%, corresponding to about 0.002 s after rounding to the precision used in Table 5. The mean of the three scenario values is 0.4301 s for PGHO and 0.4381 s for TF Greedy, so PGHO retains a small overall delay advantage while improving mean task success by about 2.62 percentage points. The comparison is useful because TF Greedy is already very competitive in instantaneous delay; PGHO adds reliability without giving up that level of mean delay performance.

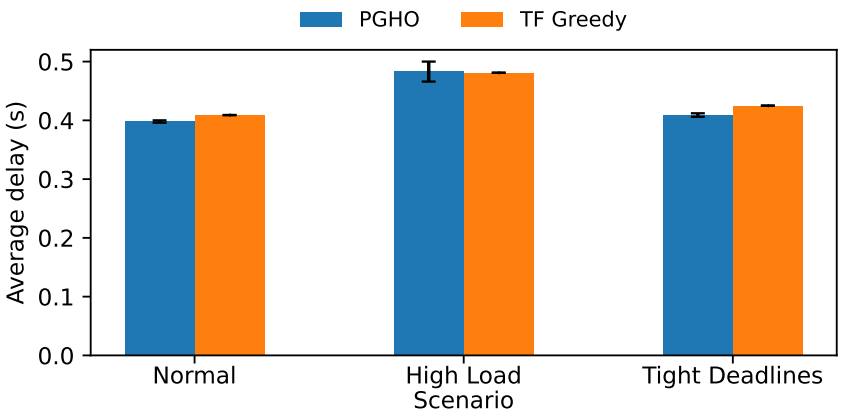


Fig. 9 Average delay comparison between TF Greedy and PGHO. Error bars indicate sample standard deviations over three random seeds.

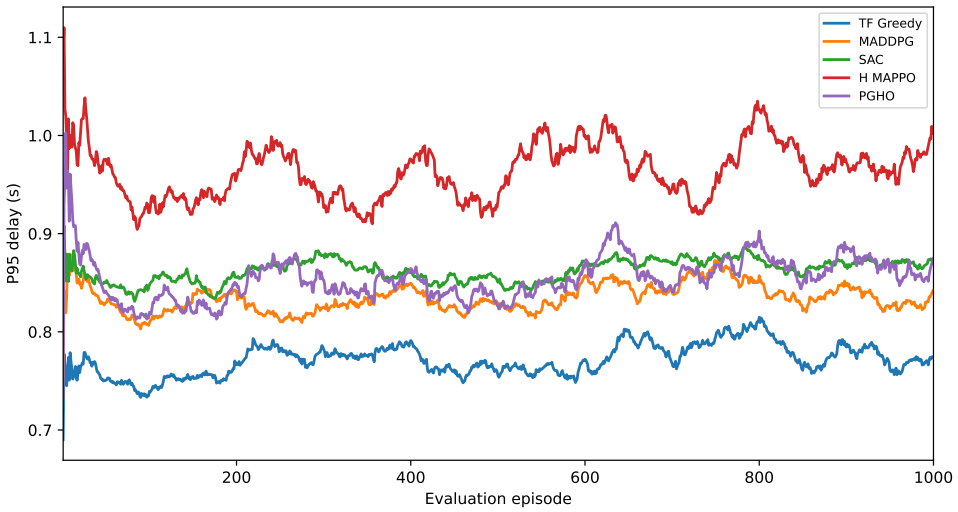


Fig. 10 P95 delay over episodes in the high load scenario

Figure 10 shows P_{95} delay over the same high load evaluation trace. PGHO is generally below SAC and H MAPPO, indicating better control of the upper part of the delay distribution than these two learning baselines. MADDPG has a slightly smaller mean P_{95} value, but its task success ratio is about 2.54 percentage points lower. The comparison reinforces the main high load result: PGHO gives up a small tail delay advantage to MADDPG with a task success ratio higher by 2.54 percentage

points while keeping the mean delay substantially lower. The curve uses the same seed aggregation and moving average as Fig. 8.

Figure 11 gives the corresponding P_{99} delay. PGHO remains generally below SAC and H MAPPO, while MADDPG retains a slightly smaller mean P_{99} value. Together with the average delay and task success ratio results, the comparison of tail delay shows that PGHO maintains low mean delay and task completion close to 100% while remaining competitive in the upper part of the delay distribution under heavy load. The curve uses the same seed aggregation and smoothing as Fig. 10.

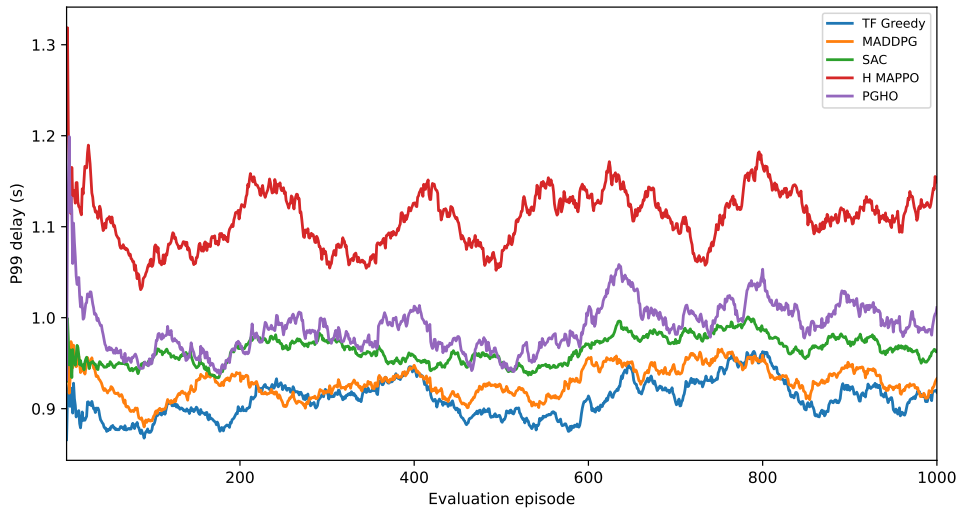


Fig. 11 P99 delay over episodes in the high load scenario

6 Conclusion

This paper presents PGHO for task offloading in VEC with UAV support and heterogeneous execution resources. PGHO summarizes several predicted task attributes as future task pressure before resource selection, then performs continuous UAV movement and offloading control under the link, queue, resource, and ratio conditions associated with the selected execution node.

Across normal load, high load, and tight deadline scenarios, PGHO achieves 0.4301 s average delay and 99.705% task success ratio. Average delay is 21.1%, 12.1%, and 7.4% lower than MADDPG, SAC, and H MAPPO, respectively. Compared with TF Greedy, PGHO keeps a small advantage in average delay while improving task success ratio by about 2.62 percentage points, at the cost of higher normalized energy cost. The component removal results show that future task information and hierarchical decision making mainly affect average and tail delay, while action masking is especially relevant to reliability under tight deadlines. The validation procedure supports policy retention during training, with elite experience and rollback serving as auxiliary operations.

The experiments use simulated traces, a single wireless model, a fixed network size, and three random seeds. Normalized energy cost does not include full UAV propulsion energy. Future work will consider measured link conditions, real vehicle trajectories, larger resource pools, prediction uncertainty, and stricter deadline settings.

References

[1] Zhao, J., Quan, H., Ge, P., Huang, Y., Xiao, Y.: Vehicular Edge Computing System: A Survey. *IEEE Internet of Things Magazine* 8(6), 122–128 (2025). https://doi.org/10.1109/MIOT.2025.3590349

[2] Meneguette, R., De Grande, R., Ueyama, J., Filho, G.P.R., Madeira, E.: Vehicular Edge Computing: Architecture, Resource Management, Security, and Challenges. *ACM Computing Surveys* 55(1), 1–46 (2023). https://doi.org/10.1145/3485129

[3] Liu, X., Liu, F., Shao, S., Li, E.: Energy-conscious task offloading for vehicular edge computing via adaptive fuzzy logic and federated learning. *Cluster Computing* 29, 778 (2026). https://doi.org/10.1007/s10586-026-06618-4

[4] Yan, M., Xiong, R., Wang, Y., Li, C.: Edge Computing Task Offloading Optimization for a UAV-assisted Internet of Vehicles via Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology* 73(4), 5647–5658 (2024). https://doi.org/10.1109/TVT.2023.3331363

[5] Dai, X., Xiao, Z., Jiang, H., Lui, J.C.S.: UAV-Assisted Task Offloading in Vehicular Edge Computing Networks. *IEEE Transactions on Mobile Computing* 23(4), 2520–2534 (2024). https://doi.org/10.1109/TMC.2023.3259394

[6] Wang, X., He, C., Jiang, W., Wang, W., Li, L., Xie, X.: UAV-Assisted Task Offloading and Resource Allocation in Internet of Vehicles: An Integration of Digital Twin and Generative AI. *IEEE Transactions on Network Science and Engineering* 13, 5038–5055 (2026). https://doi.org/10.1109/TNSE.2025.3645844

[7] Peng, P., Lin, W., Wu, W., Zhang, H., Peng, S., Wu, Q., Li, K.: A Survey on Computation Offloading in Edge Systems: From the Perspective of Deep Reinforcement Learning Approaches. *Computer Science Review* 53, 100656 (2024). https://doi.org/10.1016/j.cosrev.2024.100656

[8] He, C., Jiang, W., Wang, X., Wang, W., Wang, J., Xie, X.: Deep Reinforcement Learning and Nash Equilibrium Game-Based Task Offloading Optimization Strategy for the IoV. *IEEE Sensors Journal* 25(10), 18384–18393 (2025). https://doi.org/10.1109/JSEN.2025.3558228

[9] He, C., Jiang, W., Wang, X., Wang, W., Na, H., Xie, X.: GenAI-5GS Fusion Approach: A New Path for Enhancing Computation Offloading Efficiency of Complex Tasks in Internet of Vehicles. *IEEE Transactions on Vehicular Technology* 75(7), 12410–12425 (2026). https://doi.org/10.1109/TVT.2026.3660468

[10] Yin, L., Luo, J., Qiu, C., Wang, C., Qiao, Y.: Joint Task Offloading and Resources Allocation for Hybrid Vehicle Edge Computing Systems. *IEEE Transactions on Intelligent Transportation Systems* 25(8), 10355–10368 (2024). https://doi.org/10.1109/TITS.2024.3351635

[11] Wang, L., Wang, K., Pan, C., Xu, W., Aslam, N., Hanzo, L.: Multi-Agent Deep Reinforcement Learning-Based Trajectory Planning for Multi-UAV Assisted Mobile Edge Computing. *IEEE Transactions on Cognitive Communications and Networking* 7(1), 73–84 (2021). https://doi.org/10.1109/TCCN.2020.3027695

[12] Pateria, S., Subagdja, B., Tan, A.H., Quek, C.: Hierarchical Reinforcement Learning: A Comprehensive Survey. *ACM Computing Surveys* 54(5), Article 109, 1–35 (2021). https://doi.org/10.1145/3453160

[13] Rahmani, A.M., Haider, A., Gharehchopogh, F.S., Moghaddasi, K., Darwesh, A., Hosseinzadeh, M.: Strategic offloading in autonomous vehicles: A systematic survey of current schemes, challenges, and future prospects. *Computer Science Review* 60, 100898 (2026). https://doi.org/10.1016/j.cosrev.2026.100898

[14] Yang, L.: Task Offloading and Resource Scheduling in a Vehicle-RSU-Cloud Resource Environment. *Journal of Grid Computing* 24, Article 21 (2026). https://doi.org/10.1007/s10723-026-09843-8

[15] Sun, L., Wan, L., Wang, J., Lin, L., Gen, M.: Joint Resource Scheduling for UAV-Enabled Mobile Edge Computing System in Internet of Vehicles. *IEEE Transactions on Intelligent Transportation Systems* 24(12), 15624–15632 (2023). https://doi.org/10.1109/TITS.2022.3224320

[16] Jeong, S., Simeone, O., Kang, J.: Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning. *IEEE Transactions on Vehicular Technology* 67(3), 2049–2063 (2018). https://doi.org/10.1109/TVT.2017.2706308

[17] Long, Y., Wang, Z., Lan, S., Zhang, R., Xu, K.: Energy-latency tradeoff for task offloading and resource allocation in vehicular edge computing. *Computer Networks* 258, 111026 (2025). https://doi.org/10.1016/j.comnet.2024.111026

[18] Xie, X., Wang, H., Liu, X.: Scheduling for Minimizing the Age of Information in Multisensor Multiserver Industrial Internet of Things Systems. *IEEE Transactions on Industrial Informatics* 20(1), 573–582 (2024). https://doi.org/10.1109/TII.2023.3268749

[19] Zhang, K., Mao, Y., Leng, S., He, Y., Zhang, Y.: Mobile Edge Computing for Vehicular Networks: A Promising Network Paradigm With Predictive Offloading. *IEEE Vehicular Technology Magazine* 12(2), 36–44 (2017). https://doi.org/10.1109/MVT.2017.2668838

[20] Zeng, F., Zhang, Z., Wu, J.: Task Offloading Delay Minimization in Vehicular Edge Computing Based on Vehicle Trajectory Prediction. *Digital Communications and Networks* 11(2), 537–546 (2025). https://doi.org/10.1016/j.dcan.2024.08.003

[21] Wu, G., Miao, Y., Zhang, Y., Barnawi, A.: Energy Efficient for UAV-Enabled Mobile Edge Computing Networks: Intelligent Task Prediction and Offloading. *Computer Communications* 150, 556–562 (2020). https://doi.org/10.1016/j.comcom.2019.11.037

[22] Chhabra, G.S., Satti, S.K., Rajareddy, G.N.V., et al.: Time-and-Traffic-Aware Collaborative Task Offloading with Service Caching-Replacement in Cloud-Assisted Mobile Edge Computing. *Cluster Computing* 28, 900 (2025). https://doi.org/10.1007/s10586-025-05629-x

[23] Bo, J., Zhao, X.: Vehicle Edge Computing Task Offloading Strategy Based on Multi-Agent Deep Reinforcement Learning. *Journal of Grid Computing* 23, Article 13 (2025). https://doi.org/10.1007/s10723-025-09800-x

[24] Sun, D., Chen, Y., Li, H.: Intelligent Vehicle Computation Offloading in Vehicular Ad Hoc Networks: A Multi-Agent LSTM Approach with Deep Reinforcement Learning. *Mathematics* 12(3), 424 (2024). https://doi.org/10.3390/math12030424

[25] Luo, Q., Li, C., Luan, T.H., Shi, W., Wu, W.: Self-Learning-Based Computation Offloading for Internet of Vehicles: Model and Algorithm. *IEEE Transactions on Wireless Communications* 20(9), 5913–5925 (2021). https://doi.org/10.1109/TWC.2021.3071248

[26] Yao, L., Xu, X., Bilal, M., Wang, H.: Dynamic Edge Computation Offloading for Internet of Vehicles With Deep Reinforcement Learning. *IEEE Transactions on Intelligent Transportation Systems* 24(11), 12991–12999 (2023). https://doi.org/10.1109/TITS.2022.3178759

[27] Wu, Z., Jia, Z., Pang, X., Zhao, S.: Deep Reinforcement Learning-Based Task Offloading and Load Balancing for Vehicular Edge Computing. *Electronics* 13(8), 1511 (2024). https://doi.org/10.3390/electronics13081511

[28] Yang, W., Liu, Z., Liu, X., Ma, Y.: Deep reinforcement learning-based low-latency task offloading for mobile-edge computing networks. *Applied Soft Computing* 166, 112164 (2024). https://doi.org/10.1016/j.asoc.2024.112164

[29] Peng, H., Shen, X.: Multi-Agent Reinforcement Learning-Based Resource Management in MEC- and UAV-Assisted Vehicular Networks. *IEEE Journal on Selected Areas in Communications* 39(1), 131–141 (2021). https://doi.org/10.1109/JSAC.2020.3036962

[30] Zhao, N., Ye, Z., Pei, Y., Liang, Y.C., Niyato, D.: Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing. *IEEE Transactions on Wireless Communications* 21(9), 6949–6960 (2022). https://doi.org/10.1109/TWC.2022.3153316

[31] Ju, T., Li, L., Liu, S., Zhang, Y.: A Multi-UAV Assisted Task Offloading and Path Optimization for Mobile Edge Computing via Multi-Agent Deep Reinforcement Learning. *Journal of Network and Computer Applications* 229, 103919 (2024). https://doi.org/10.1016/j.jnca.2024.103919

[32] Liu, Y., Shi, X., Liu, K., He, Y., Wang, H.: Queue-Stable Dynamic Deployment for UAV-Assisted Edge Networks via Lyapunov-Guided Deep Reinforcement Learning. *Journal of Grid Computing* 24, Article 27 (2026). https://doi.org/10.1007/s10723-026-09848-3

[33] Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., Mordatch, I.: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: *Advances in Neural Information Processing Systems*, vol. 30, pp. 6379–6390 (2017).

[34] Zakaryia, S.A., Mead, M.A., Nabil, T., et al.: Task Offloading for Multi-UAV Asset Edge Computing with Deep Reinforcement Learning. *Cluster Computing* 28, 462 (2025). https://doi.org/10.1007/s10586-025-05382-1

[35] Liu, J., Zou, Y., Du, G., Zhang, X., Wu, J.: Task Offloading and Resource Allocation for ICVs in Vehicular Edge Computing Networks Based on Hybrid Hierarchical Deep Reinforcement Learning. *Sensors* 25(22), 6914 (2025). https://doi.org/10.3390/s25226914

[36] Du, J., Chen, T., Jiang, J., Chen, Y., Zhao, H., Gong, Y., Gui, G.: Hierarchical Deep Reinforcement Learning for Task Failure Minimization in MEC-Enabled TDMA-Based Internet of Vehicle Networks. *IEEE Transactions on Vehicular Technology*, 1–6 (2026). https://doi.org/10.1109/TVT.2026.3728791

[37] Yang, Q., Yoo, S.-J.: Adaptive UAV-Aided Fair Task Offloading With Fuzzy Caching for Vehicular Edge Computing via Hierarchical Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology* 75(6), 11558–11574 (2026). https://doi.org/10.1109/TVT.2025.3647844

[38] Navidi, S.M., Rezvani, M.H., Khosraghi, E.Z.: Optimizing Computation Offloading in IoV Environments: A Hybrid Learning Automata and Deep Recurrent Q-Network Approach. *Journal of Grid Computing* 24, Article 20 (2026). https://doi.org/10.1007/s10723-026-09841-w

[39] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep Reinforcement Learning That Matters. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 3207–3214 (2018). https://doi.org/10.1609/aaai.v32i1.11694

[40] Oh, J., Guo, Y., Singh, S., Lee, H.: Self-Imitation Learning. In: *Proceedings of the 35th International Conference on Machine Learning, Proceedings of Machine Learning Research*, vol. 80, pp. 3878–3887 (2018).

[41] You, C., Huang, K., Chae, H., Kim, B.H.: Energy-Efficient Resource Allocation for Mobile-Edge Computation Offloading. *IEEE Transactions on Wireless Communications* 16(3), 1397–1411 (2017). https://doi.org/10.1109/TWC.2016.2633522

[42] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In: *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, vol. 80, pp. 1861–1870 (2018).

[43] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y.: The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. In: *Advances in Neural Information Processing Systems*, vol. 35, pp. 24611–24624 (2022).

[44] Cai, S., Xiao, J., Cao, Y., Gao, Q., Feng, Z., Deng, S.: Dynamic Task Transmission Control and Improved Greedy Strategy for Vehicular Edge Computing. *Future Generation Computer Systems* 180, 108369 (2026). https://doi.org/10.1016/j.future.2026.108369

[45] Liu, J., Wang, Y., Pan, D., Yuan, D.: QoS-Aware Task Offloading and Resource Allocation Optimization in Vehicular Edge Computing Networks via MADDPG. *Computer Networks* 242, 110282 (2024). https://doi.org/10.1016/j.comnet.2024.110282

[46] Zhao, X., Zhao, T., Wang, F., Wu, Y., Li, M.: SAC-Based UAV Mobile Edge Computing for Energy Minimization and Secure Data Transmission. *Ad Hoc Networks* 157, 103435 (2024). https://doi.org/10.1016/j.adhoc.2024.103435

[47] Zhang, J., Niu, Y., Lin, Z., Yang, J., Zhang, W.: Load-Balancing Multi-UAV Collaborative MEC Offloading-Migration and Trajectory Optimization Based on Lyapunov MAPPO. *Computer Communications* 257, 108605 (2026). https://doi.org/10.1016/j.comcom.2026.108605

Statements and Declarations

Author contributions. Chao He: Supervision, Resources, Project administration, Funding acquisition, conceptual discussion, methodological refinement, Writing–review & editing. Dongfeng Fu: Conceptualization, Methodology, Software, Investigation, Formal analysis, Validation, Visualization, Writing–original draft, Writing–review & editing. Xin Xie: Investigation, Formal analysis, Writing–review & editing.

Jinkui Zhang: Resources, Writing–review & editing. ShuXu Yue: Investigation, Visualization, Writing–review & editing. Zheng Zhang: Project administration, Writing–review & editing. All authors reviewed and approved the final version of the manuscript.

Funding. This work was supported by the National Natural Science Foundation of China (Grant No. 62402081), the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant Nos. KJZDK202201203, KJQN202301258, and KJQN202400645), the Natural Science Foundation of Chongqing, China (Grant No. CSTB2025NSCQ-GPX0246), the Green Ecological Water Conservancy and Electric Power Engineering Research Center, Yunnan Water Resources and Hydropower Vocational College (Grant No. 2026YSZSYS006), and the Yunnan Provincial Engineering Research Center for Green Energy and Smart Power Grid Projects (Grant No. 2026YSZSYS026).

Competing interests. The authors have no relevant financial or non-financial interests to disclose.

Ethics approval and consent to participate. Not applicable.

Consent for publication. Not applicable.

Data availability. The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.

Materials availability. Not applicable.

Code availability. The source code is not publicly available.