ARTICLE | Article in Press

Graph neural network-based service migration decision in vehicular networks

Liyan Wang · Xianfeng Zheng · Liang Liu · Hao Feng · Wenwei Li · Huan Liu

Received: 10 October 2025 / Accepted: 11 September 2026

Abstract

The rapid development of the Internet of Vehicles (IoV) has led to an exponential increase in the number of latency sensitive and computationally intensive tasks. Due to the limitations of onboard computing resources in vehicles, offloading these tasks generated during vehicle operation to a remote cloud for processing inevitably leads to significant transmission delays. By combining the IoV with Mobile Edge Computing (MEC), these latency sensitive and computationally intensive tasks can be processed in MEC server to reduce energy consumption of vehicles and the transmission delay of tasks. However, due to the small coverage area of MEC servers and the mobility of vehicles, it may cause vehicles using MEC services to leave the coverage area of current MEC servers, resulting in service interruptions. How to dynamically migrate tasks for ensuring service continuity is a great challenge. In this paper, we study service migration strategies in dynamic and complex IoV MEC environments. We first model the service migration strategies as a multi-objective optimization problem that minimizes the weighted sum of latency and energy consumption. We then transform the problem into a Markov Decision Process (MDP). To capture time-varying server loads and link-state features over the physical RSU graph, a Graph Convolutional Network (GCN) is employed to extract features from the network topology and node information in the vehicular edge network, and a graph convolutional neural network-based deep reinforcement learning task migration decision algorithm (Gr-MiD) is proposed to make service migration decisions based on task details. Simulation results in a small synthetic intra-zone IoV-MEC setting demonstrate that the Gr-MiD algorithm outperforms existing comparative algorithms, achieving an 8.04% reduction in average task delay and a 15.82% reduction in average task energy consumption compared to the DQN baseline under peak-load conditions.

Keywords Internet of vehicle · Service migration · Latency and energy consumption · Graph neural network

These authors contributed equally: Liyan Wang, Xianfeng Zheng, Hao Feng and Wenwei Li.

https://doi.org/10.1038/s41598-026-71909-0

Published online: 19 September 2026

scientific reports

Introduction

The rapid advancements in Artificial Intelligence (AI) and the Internet of Vehicles (IoV) have led to the emergence of intelligent automotive applications such as autonomous driving and collision warning systems[1–3]. However, processing these compute-intensive and latency-sensitive tasks poses a significant challenge for resource-limited onboard devices. Although tasks can be offloaded to remote cloud servers with abundant computing resources[4], the high latency caused by long-distance data transmission fails to meet the requirements for latency-sensitive tasks in vehicles[5–7]. To address this issue, researchers have introduced Mobile Edge Computing (MEC) technology into the IoV. By deploying MEC servers in Road Side Units (RSUs) along the roads, forming vehicular edge computing, MEC provides short-range task offloading services for vehicles[8–10]. Offloading latency-sensitive or computation-intensive tasks to nearby MEC servers via the IoV can extend the computing and storage capabilities of vehicular devices, thereby meeting the computational and storage resource demands of vehicle users[11–13].

Although offloading tasks to MEC servers offers advantages in reducing both the energy consumption of onboard devices and task response latency, it introduces new challenges. The limited service coverage of MEC servers and the high-speed random movement of vehicles in IoV can result in situations where the offloaded service is not yet completed when the vehicle moves out of the current MEC server's coverage area. This can cause high service latency, instability, or even service interruptions[14–18]. Therefore, it is a significant challenge to ensure efficient task migration in a dynamically complex IoV-MEC network environment to maintain service continuity.

In vehicle networks, the time-varying vehicle associations, server loads, and link-state features are represented over a physical RSU graph. The state of each node (e.g., MEC servers) depends on its intrinsic resources (such as computing capacity and load) as well as the topological linkages of neighboring nodes (e.g., the communication bandwidth between adjacent RSUs). Graph Convolutional Network (GCN) aggregate features from neighboring nodes via propagation over the graph structure, enabling effective capture of these non-Euclidean spatial dependencies. This mechanism provides a more comprehensive feature representation for migration decisions under dynamic network states.

Currently, some studies have been conducted on service migration issues, Kim et al.[19] designs a system for optimizing edge computing deployment and proposes a distributed heuristic algorithm based on user location prediction to optimize task migration in MEC. However, this study does not address the problem of server loads and link-state features changing due to vehicle mobility, and the method, which relies on location-based predictive task migration, can lead to reduced resource utilization. Additionally, some research employs traditional deep reinforcement learning algorithms to address service migration decision problems. Due to factors such as vehicle movement, varying MEC server resource loads, or MEC server failures, the network state in IoV continuously changes, leading to challenges such as high dimensionality of data and action space, and difficulty in model convergence with traditional deep reinforcement learning algorithms. Furthermore, the robustness of these algorithm models is low in scenarios with time-varying IoV network states.

Different from the above studies, this paper investigates the service migration decision-making problem in dynamic vehicular network environments based on GCN. Firstly, we establish models for vehicle user migration latency and energy consumption, framing the service migration decision problem as a multi-objective optimization problem aiming to minimize the weighted sum of task migration latency and energy consumption. Secondly, considering the dynamic changes in vehicle associations, link states, and resources in vehicular networks, we design a task migration decision algorithm based on GCN to ensure the continuity of vehicle user services. Specifically, the MEC controller monitors the state of the physical network and its topology, and the GCN is used to extract features from edge node information. Using these extracted edge node features and the task characteristics of vehicles, a Deep Reinforcement Learning (DRL) algorithm is employed to make vehicle service migration decisions. Finally, simulations are conducted to evaluate the performance of the proposed algorithm. The main contributions of this paper are as follows:

- We define and model the service migration problem in IoV-MEC networks as a multi-objective optimization problem, aimed at minimizing the weighted sum of latency and energy consumption. We further model this problem as a Markov Decision Process (MDP).
- We utilize a Graph Convolutional Network (GCN) to extract features from the network topology and resource information of edge nodes in the IoV-MEC environment. This allows the feature information on edge nodes to aggregate the characteristics of neighboring nodes and links, thereby better capturing the dynamic nature of the IoV-MEC network.
- We design a Graph Convolutional Network-based Deep Reinforcement Learning Task Migration Decision algorithm (Gr-MiD). This algorithm can adaptively make service migration decisions for tasks in a dynamic IoV-MEC environment.
- We conduct simulation experiments in a small synthetic intra-zone IoV-MEC setting (a single MEC-controller zone with $K = 5$ servers and Gaussian-Markov mobility), and the results demonstrate that the proposed algorithm outperforms the compared baseline algorithms in terms of reducing latency and energy consumption.

The remainder of this paper is organized as follows. Section "Related work" presents the related work. The system model and the problem definition are described in Sect. "System model and problem definition" followed by a detailed discussion of the service migration problem. In Sect. "Migration algorithm model", we develop the Graph Neural Network-based algorithm. The performance of the proposed approaches is analyzed in Sect. "Performance evaluation" and the conclusions are made in Sect. "Conclusion".

Related work

A significant amount of research has been conducted on the service migration problem in MEC environments.

Liu et al.[20] proposed a Latency-aware Service Migration method with Decision theory (LSMD) to optimize service latency and balance the workload of roadside units, which overlooks the impact of resource limitations of edge devices and complex network environments on service migration. Abouaomar et al.[21] targeted minimizing total service latency and migration costs and investigated the service migration problem in MEC-supported vehicular networks. The authors modeled the service migration problem as a multi-agent Markov Decision Process and addressed it using Deep Q-Networks (DQN). Zhou et al.[22] introduced a service composition and Virtual Machine (VM) migration scheme in cloud environments to reduce network resource consumption in data centers. An optimization objective function was established based on the system's load under certain states, and an optimizer was used to solve the task migration decision problem. Wang et al.[23] explored joint task offloading and migration schemes in MEC scenarios, proposing a reinforcement learning-based service migration strategy to maximize system benefits. However, the above studies[21–23] do not account for dynamic network changes. In the context of vehicular networks, the mobility of vehicles and the dynamism of onboard tasks lead to dynamic environmental changes, making the aforementioned methods unsuitable for handling task migration in IoV-MEC environments.

Li et al.[24] investigated a novel mobile-aware dynamic service placement and migration problem in Device-to-Device (D2D) communications. Specifically, it addressed how to optimize service offloading and migration between devices under conditions of device mobility, aiming to minimize task offloading costs, including computation costs, communication latency costs, and migration costs. The authors proposed a mechanism to optimize total delay and energy costs for dynamic service offloading and migration in D2D systems without prior knowledge of future mobility information. Wang et al.[25] studied the service migration decision problem in vehicular networks where services can be migrated through Vehicle-to-Vehicle (V2V) or Infrastructure-to-Infrastructure (I2I) channels. The authors proposed a migration algorithm based on game theory to minimize task migration energy consumption and latency. However, these studies employ traditional algorithms to solve the task migration problem, which may encounter issues such as computational explosion.

Authors in[26] and[27] have explored the service migration problem in dynamic vehicular networks. Specifically, Chen et al.[26] investigated service migration in MEC-supported vehicular networks with the goal of minimizing total service delay and migration costs. The authors employed Double Deep Q-Networks (DDQN) while considering factors such as vehicle mobility and server load. Zhang et al.[27] proposed a mobile-aware service migration mechanism based on Deep Reinforcement Learning (DRL) in vehicular networks to effectively reduce service and migration delays. This approach considered edge network states and user mobility information to ensure real-time service migration decisions. Luo et al.[28] proposed a joint task migration and resource allocation approach based on Deep Deterministic Policy Gradient (DDPG) in vehicular edge computing to minimize the average weighted cost. This approach considered vehicle mobility, task execution time, and the costs of using edge server resources and transmitting tasks between RSUs. However, these studies did not address the issue of energy consumption in service migration, and traditional DRL algorithms may fall into local optima when faced with the dynamically changing edge vehicular network environment. To facilitate comparison, the core evaluation metrics and key performance data of the above methods are summarized in Table 1.

To overcome these limitations of conventional DRL-based approaches in handling dynamic network topologies, recent research has demonstrated the effectiveness of graph-based optimization methods in vehicular networks. Authors in[29] proposed the temporal graph convolutional network model that combines GCNs and gated recurrent units to simultaneously capture spatial topological dependencies and temporal dynamics in urban road networks for accurate real-time traffic forecasting. The authors in[30] proposed vehicular federated graph neural network, a computation offloading mechanism based on federated graph neural networks for vehicular networks that considers vehicle characteristics and network topology to minimize weighted computation offloading delays, achieving high prediction accuracy and low offloading latency. Ullah et al.[31] study proposed a task offloading scheme for vehicular edge computing that integrates GCNs with Double-DQN to enable vehicles to dynamically choose between local execution or offloading to nearby vehicles, edge servers, or cloud while simultaneously minimizing cost and delay through a Markov Decision Process optimization framework. Luo et al.[32] proposed a Causality-Aware Graph Convolutional Network (CA-GCN) for vehicular network anomaly detection that learns causal relationships among message semantics to improve detection accuracy and reduce false positives. These studies collectively demonstrate the significant potential of applying graph-based approaches to service migration decision-making in dynamic IoV-MEC environments, where capturing the complex interdependencies between network topology, vehicle mobility, and resource allocation is crucial for optimal performance.

To address the limitations of existing work, it is worth clarifying how the proposed approach differs from the graph-based methods discussed above. Prior graph-RL works[30,31] focus on offloading destination selection among multiple candidate servers, whereas this paper addresses a structurally distinct problem: the choice between active and passive migration schemes, where the cost trade-off between service instance transfer and remote data forwarding must be explicitly modeled. Moreover, existing graph-based methods typically optimize a single objective, while this work jointly minimizes latency and energy consumption. We further adopt an Actor-Critic policy gradient framework rather than value-based methods, as it offers more stable convergence under the non-stationary, frequently-triggered migration events characteristic of dynamic IoV environments. These distinctions collectively motivate the design of Gr-MiD.

Table 1 Performance metrics and results comparison of existing studies.

Literature	Method	Core evaluation metrics	Key performance data
[19]	Heuristic Algorithm	User latency	User latency is lowered by 33% relative to earlier methodologies
[25]	Game Theory	Computation overhead	Achieves a maximum reduction rate of 24.6% when the data size is 15 Mbit.
[26]	DDQN	Average long-term cost	30% lower than DQN
[27]	DQN	Average migration delay	The average migration delay leads by 12%-30%
[28]	DDPG	Average weighted cost	The average weighted cost is reduced by approximately 8%-20%

System model and problem definition

As illustrated in Fig. 1, the considered system model comprises R different vehicle devices, U MEC controllers (each covering K RSUs), and a central cloud server.We denote the sets of vehicles, MEC controllers, and RSUs as $\mathcal{R} = \{1, 2, \cdots, R\}$,$\mathcal{U} = \{1, 2, \cdots, U\}$, and $\mathcal{K} = \{1, 2, \cdots, K\}$, respectively. The RSUs are uniformly distributed along the roadside, with each RSU housing an MEC server. Vehicle devices can access and utilize the computing resources of both MEC servers and the central cloud server. The designed structure consists of three distinct layers: the central cloud layer, the edge layer, and the vehicle device layer.

In this system, the edge layer is divided into multiple zones, with each zone deploying an MEC controller responsible for managing the MEC servers in the RSUs within that zone. Task offloading and initial service placement are assumed to have been completed before a vehicle handoff occurs. At the start of the task migration decision process, the vehicle device uploads its feature information and task details to the MEC controller. The MEC controller gathers real-time feature information from the RSUs within its coverage area and subsequently determines the migration strategy through the migration decision model.

Edge nodes can communicate with each other through Infrastructure-to-Infrastructure (I2I) connections using wired links. Therefore, when active migration is selected, the service instance can be transferred to the new serving MEC server through these communication links. When a vehicle handoff triggers a migration decision, the controller collects feature information about the MEC servers and the task. The migration decision model located in the MEC controller then chooses either active migration or passive migration.

Services model

In this system model, each MEC server has computing resources C_k and storagere sourcesD_k. The feature information of the MEC server located in the k-th RSU is denoted as $\{D_k, C_k, L_k\}, 1 \leq k \leq K$, where L_k represents the set of tasks waiting to be processed on the k-th MEC server. In the IoV context, a vehicle may need to execute a series of complex tasks. For example, in an intelligent transportation system scenario, vehicle devices may need to perform tasks including but not limited to adaptive cruise control, traffic sign recognition, and automatic parking. Each task may require different processing capabilities and response times, and have varying demands on resource utilization and latency. To ensure that these tasks are completed efficiently and timely, they can be decomposed into smaller, independent sub-tasks and optimized individually. This approach allows multiple sub-tasks to be executed in parallel, thereby reducing the overall system response time and improving vehicle operation efficiency. In this paper, it is assumed that a vehicle's task can be decomposed into N independent sub-tasks, defined as follows:

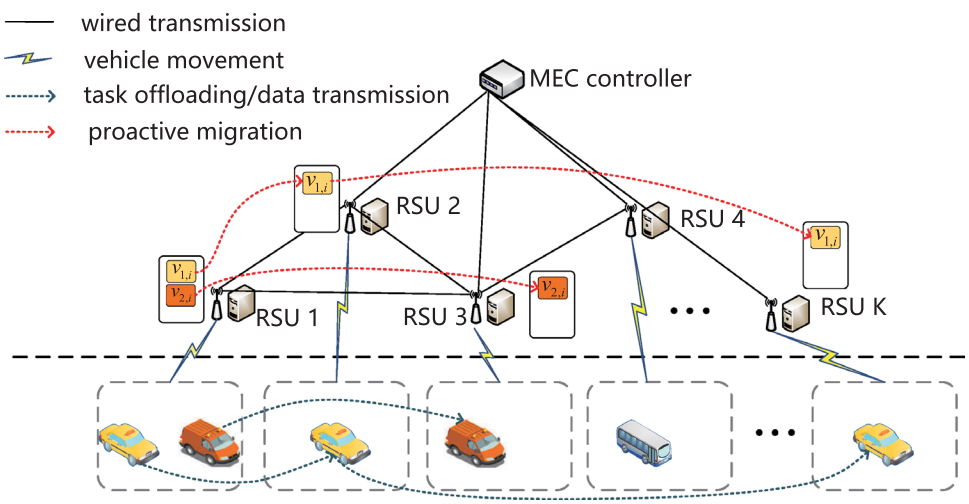

Fig. 1 The service migration system model.

$$\mathcal{V}_r = \{v_1, v_2, \ldots, v_N\} \tag{1}$$

where $\mathcal{V}_r$ denotes the set of N independent sub-tasks of the task on the r-th vehicle, $1 \le r \le R$, and v_i $(1 \le i \le N)$ denotes its i-th sub-task. The feature information of v_i is $\{d_{r,i}, c_{r,i}, t_{r,i}^{\max}\}$, where $d_{r,i}$ is the task data volume, $c_{r,i}$ is the computational workload measured in CPU cycles, and $t_{r,i}^{\max}$ is the maximum acceptable processing delay. The processing rate of MEC server k is denoted by C_k and is measured in cycles/s. Vehicle devices access MEC servers through RSU-based V2I links, consistent with mainstream vehicular edge-computing deployments[8], providing a tractable baseline for future extension to multi-RAT scenarios.

Migration model

In the dynamically changing IoV environment, the position of a vehicle changes in each time slot $t \in \{0, 1, 2 \cdots T\}$. Let $E_t^s(r) \in K$ denote the edge node to which vehicle r is connected at time slot t. By comparing the edge nodes connected by the vehicle in adjacent time slots, the migration trigger state X_r of vehicle r can be defined as follows:

$$X_r = \begin{cases} 0, E_t^s(r) = E_{t-1}^s(r) \\ 1, E_t^s(r) \neq E_{t-1}^s(r) \end{cases} \tag{2}$$

When $X_r = 0$, it indicates that the vehicle is moving within the coverage area of the MEC server and does not require a service migration decision. When $X_r = 1$, it signifies that the vehicle has crossed into a different coverage area, triggering a migration event, and a service migration decision needs to be made. Let $\rho(r, i)$ denote the migration decision for task v_i of vehicle r at the time of migration trigger. When $\rho(r, i) = 1$, it indicates an active migration approach, where the vehicle migrates the service instance to a new MEC server for processing. When $\rho(r, i) = 0$, it indicates a passive migration approach, where the current MEC server does not migrate the task instance, and the vehicle transfers task data to the original MEC server through RSU-to-RSU communication links.Both approaches incur different costs: the passive migration method increases communication costs due to data forwarding through RSUs, while the active migration method incurs migration costs due to the actual service migration. Therefore, service migration strategies should be formulated considering both latency and energy consumption costs. Here, X_r functions purely as a migration event detector rather than a migration executor. The actual strategy is determined by the decision variable $\rho(r, i)$: selecting passive migration ($\rho(r, i) = 0$) retains the task on the original MEC server until completion, which functionally subsumes scenarios where migration is inadvisable due to advanced task execution progress. Furthermore, the temporal uncertainty of vehicle dwell time within the current RSU coverage area is handled implicitly through the MDP formulation: the migration trigger condition X_r is evaluated at each discrete time slot t, and the policy π_θ is trained over diverse vehicle mobility traces generated by the Gaussian-Markov model, enabling the actor network to learn temporally-aware migration policies without requiring an explicit dwell-time predictor.

Latency model

In vehicular networks, the service migration process primarily involves the following steps: the vehicle device uploads the task to the MEC server, the service instance corresponding to the vehicle migrates between MEC servers, the MEC server processes the task from the vehicle device, and the results are returned to the vehicle device via the RSU. Therefore, the considered delay model includes the delay for the vehicle device to upload the task to the MEC server, the migration delay of the service instance between MEC servers, the queuing delay of the task on the MEC server, the processing delay of the vehicle task on the MEC server, and the return delay for the RSU to transmit the computed results back to the vehicle device.

Since the vehicle device and RSU use a wireless network for bidirectional data transmission, the upload rate of tasks from the vehicle device to the MEC server is influenced by the vehicle's transmission power, the communication link bandwidth, the channel gain between the vehicle and RSU, and the noise power in the channel.

According to Shannon's theorem, the task transmission rate from the vehicle device to the MEC server can be expressed as:

$$R_{V2E} = B_1 \cdot \log_2\left(1 + \frac{p \cdot h_{r,k}}{\sigma^2}\right) \tag{3}$$

where B_1 represents the bandwidth of the communication link between the vehicle and the RSU, p is the transmission power of the vehicle, Let $h_{r,k}$ denote the channel gain between vehicle r and MEC server k, σ^2 denotes the power of additive white Gaussian noise in the channel.

The upload latency depends solely on the transmission rate and the size of the task data. Thus, the delay for uploading a task from the vehicle device to the MEC server can be expressed as:

$$t_{r,i}^{up} = \frac{d_{r,i}}{R_{V2E}} \tag{4}$$

The processing latency depends on the task workload $c_{r,i}$ (cycles) and the processing rate C_k (cycles/s) of the MEC server on which the sub-task is executed. Under the sequential FCFS service model, the processing delay is

$$t_{r,i}^{proc} = \frac{c_{r,i}}{C_k}. \tag{5}$$

Let $\mathcal{P}_k(r,i;t)$ denote the set of unfinished tasks that arrived at server k before task (r, i) at time t. The FCFS queuing delay includes only these preceding tasks and is therefore defined as

$$t_{r,i}^{que} = \sum_{(r',j)\in\mathcal{P}_k(r,i;t)} \frac{c_{r',j}}{C_k}, \tag{6}$$

where the sum is zero when $\mathcal{P}_k(r,i;t) = \emptyset$. This formulation excludes tasks arriving after (r, i) and is dimensionally expressed in seconds.

Due to the varying delays caused by different migration schemes, when a vehicle triggers migration and uses the active migration scheme, the delays to consider include: the delay for the vehicle device to upload the task to the MEC server, the queuing delay for the vehicle task at the MEC server, the processing delay for the vehicle task at the MEC server, and the migration delay of service instances between MEC servers. Since the data volume of the processing result is very small, the return delay is not considered. The migration delay of service instances between MEC servers approaches a constant value, which is denoted as t_m. This constant approximation reflects the container-level migration paradigm adopted in this work. Unlike VM migration, which transfers entire OS images and incurs minute-level delays, container migration exploits lightweight image layering to achieve second-level migration times[33]. Since container image sizes remain stable within the same IoV-MEC zone, both t_m and e_{mar} are reasonably modeled as scenario-averaged constants. Therefore, the total delay incurred using the active migration scheme can be expressed as:

$$t_{r,i}^{Act} = t_{r,i}^{up} + t_{r,i}^{que} + t_{r,i}^{proc} + t_m \tag{7}$$

When adopting the passive migration scheme, the delays to consider include: the delay for the vehicle device to upload the task to the MEC server, the queuing delay for the vehicle's task on the MEC server, the processing delay of the vehicle's task on the MEC server, and the communication delay between two edge servers for task transmission. Therefore, the total delay incurred under the passive migration scheme can be expressed as:

$$t_{r,i}^{Pas} = t_{r,i}^{up} + t_{r,i}^{que} + t_{r,i}^{proc} + \frac{d_{r,i}}{R_{E2E}} \tag{8}$$

where R_{E2E} represents the transmission rate of task data between edge servers. Similarly, this can be expressed using the Shannon formula as:

$$R_{E2E} = B_2 \cdot \log_2\left(1 + \frac{p_e \cdot h_{l,s}}{\sigma^2}\right) \tag{9}$$

where B_2 denotes the link bandwidth between MEC servers, p_e is the transmitting power of the RSUs, $h_{l,s}$ denotes the channel gain between RSUs, and σ^2 represents the additive Gaussian white noise power in the channel.

Energy consumption model

In the task migration phase, the main considerations for energy consumption include: the energy consumed during transmission and the energy consumed by the MEC servers for task processing.

The energy consumed during transmission is divided into two types: one is the energy consumed when the task is uploaded to the MEC server, which is primarily related to the vehicle's transmission power and the task upload delay. This can be expressed as:

$$e_{r,i}^{tran,up} = p \cdot t_{r,i}^{up} \tag{10}$$

The other type is the energy consumed during the migration of tasks between MEC servers, which can be expressed as:

$$e_{r,i}^{tran,mar} = e_{E2E} \cdot d_{r,i} \tag{11}$$

where e_{E2E} represents the energy consumed for transmitting each unit of data between MEC servers.

The processing energy consumption is expressed as:

$$e_{r,i}^{proc} = e_{edge} \cdot c_{r,i} \tag{12}$$

where e_{edge} denotes the energy consumed per CPU cycle on the MEC server.

Similar to the latency model, different task migration schemes result in different energy consumption. When the proactive migration scheme is chosen, the energy consumption to consider includes: the energy consumed during the upload of tasks to the MEC server, the processing energy consumed on the MEC server, and the migration energy incurred when transferring service instances between MEC servers. It is important to note that the migration energy between MEC servers tends to be a constant, denoted as e_{mar}. Therefore, the total energy consumption for the proactive migration scheme can be expressed as:

$$e_{r,i}^{Act} = e_{r,i}^{tran,up} + e_{r,i}^{proc} + e_{mar} \tag{13}$$

When using the passive migration scheme, the main energy consumption to consider includes: the energy consumed during task upload to the MEC server, the energy consumed during task data transfer between MEC servers, and the energy consumed during task processing on the MEC server. Therefore, the total energy consumed when executing the passive migration scheme can be expressed as:

$$e_{r,i}^{Pas} = e_{r,i}^{tran,up} + e_{r,i}^{proc} + e_{r,i}^{tran,mar} \tag{14}$$

Problem formulation

Based on the introduced migration decision variables, the latency incurred by sub-task v_i of vehicle r under decision $\rho(r,i)$ can be expressed as:

$$t_{r,i}^{total} = \rho(r,i) \cdot t_{r,i}^{Act} + [1 - \rho(r,i)] \cdot t_{r,i}^{Pas} \tag{15}$$

Similarly, the energy consumption incurred by sub-task v_i of vehicle r is:

$$e_{r,i}^{total} = \rho(r,i) \cdot e_{r,i}^{Act} + \left[1 - \rho(r,i)\right] \cdot e_{r,i}^{Pas} \tag{16}$$

Let T_{ref} and E_{ref} denote the per-task reference latency and energy, respectively. Because each migration decision $\rho(r,i)$ is made independently per sub-task, the vehicle-level objective is the sum of the per-task normalized costs, each normalized by the same per-task reference. The utility function is

$$U = \sum_{i=1}^{N} \left(\varepsilon_1 \frac{t_{r,i}^{total}}{T_{\text{ref}}} + \varepsilon_2 \frac{e_{r,i}^{total}}{E_{\text{ref}}} \right). \tag{17}$$

Therefore, the optimization problem is formulated as

$$\begin{aligned} &\underset{\rho(r,i)}{Minimize} : U = \sum_{i=1}^{N} \left(\varepsilon_1 \frac{t_{r,i}^{total}}{T_{\text{ref}}} + \varepsilon_2 \frac{e_{r,i}^{total}}{E_{\text{ref}}} \right) \\ &S.t. \begin{cases} C1 : \varepsilon_1 + \varepsilon_2 = 1, \varepsilon_1 \in [0,1], \varepsilon_2 \in [0,1] \\ C2 : \sum\limits_{(r,i) \in \mathcal{A}_k(t)} c_{r,i} \le C_k \Delta t \\ C3 : \sum\limits_{(r,i) \in L_k(t)} d_{r,i} \le D_k \\ C4 : \rho(r,i) \in \{0,1\} \end{cases} \end{aligned} \tag{18}$$

Here, ε_1 and ε_2 specify the relative preference between the two dimensionless per-task objectives, and T_{ref} and E_{ref} are defined at the same per-task aggregation level as the normalized variables in both the objective and the reward defined in Eq. (21). In constraint $C2$, $\mathcal{A}_k(t)$ denotes the set of sub-tasks newly admitted to server k during scheduling interval Δt; the workload of the newly admitted sub-tasks, $c_{r,i}$ (cycles), must not exceed the processing budget $C_k \Delta t$ (cycles) available in that interval. Sub-tasks already queued on server k are not re-counted in $C2$, so a legitimate multi-slot FCFS backlog may span several intervals. Constraint $C3$ limits the total data volume of the queued backlog $L_k(t)$ to the available storage capacity D_k, and $C4$ defines the binary migration-scheme decision.

Migration algorithm model

To address the optimization objective presented in Eq. (18), this paper proposes a Graph Convolutional Network-based Deep Reinforcement Learning Task Migration Decision Algorithm (Gr-MiD) to determine the optimal task migration strategy in vehicular networks. As illustrated in Fig. 2, the proposed task migration decision model first employs a Graph Convolutional Network (GCN) to extract topology-aware features from the edge-node information, including both node resource states and inter-node link information. In the considered experimental setting, each MEC-controller zone contains K=5 MEC servers, which is used to instantiate the graph-related tensor dimensions described in the following sections. The extracted graph representation is then integrated with the task feature information to construct the state representation used by the Actor–Critic framework. Finally, the Deep Reinforcement Learning (DRL) algorithm generates migration decisions aimed at minimizing the weighted sum of task migration delay and energy consumption.

The Actor-Critic framework is adopted over value-based alternatives because frequent and irregular migration triggers in IoV environments create non-stationary state distributions, under which policy gradient methods offer more stable on-policy updates than Q-value approximation. The GCN is incorporated as the state-representation backbone. Its node embeddings are first aggregated by the global mean-pooling readout and projected, which is then concatenated with the task features to form the state supplied to the Actor–Critic framework.

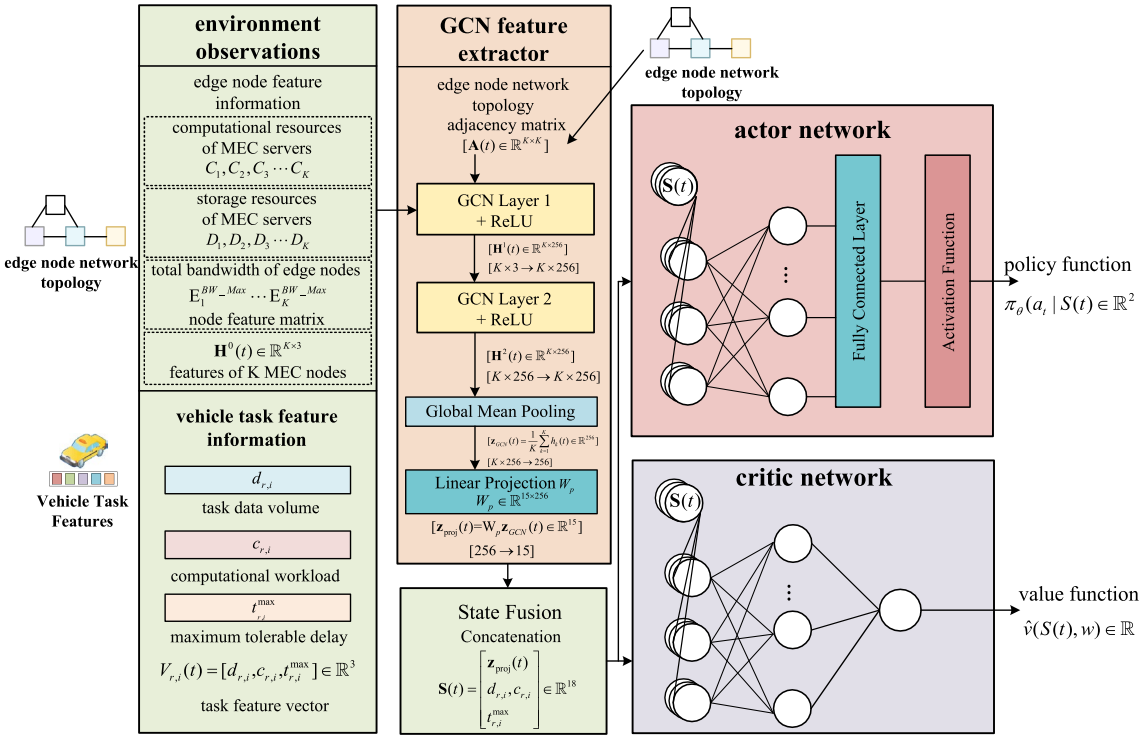

Fig. 2 Migration decision algorithm model based on GCN. The node feature matrix $\mathrm{H}^0 \in \mathbb{R}^{K\times C}$ and the adjacency matrix A are processed by the GCN into topology-aware embeddings $\mathrm{H}^2 \in \mathbb{R}^{K\times F'}$. A mean-pooling readout followed by a linear projection reduces H^2 to a K-invariant vector $\mathrm{z}_{\mathrm{proj}} \in \mathbb{R}^{15}$, which is concatenated with the task features $v_{r,i}$ to form the 18-dimensional state; the Actor network then outputs the binary migration decision $\rho_{r,i}(t)$.

Reinforcement learning environment

The reinforcement learning framework consists of three main components: state, action, and reward. Each of these components is described in detail below.

State Space: In task migration, it is essential to consider the characteristics of MEC servers, the link information between MEC servers, and the task characteristics. Although link states might be useful in task migration within vehicular networks, the number of links typically exceeds the number of nodes in such environments. For simplicity and computational efficiency, this paper integrates link information into the node feature information, represented as $\mathrm{E}_k^{BW\_Max}$, which can be understood as the total bandwidth of all links connected to edge node k. These per-server features $\{D_k, C_k, \mathrm{E}_k^{BW\_Max}\}$ are collected into the node feature matrix H^0 (Eq. (22)) and processed by the GCN. To obtain a fixed-dimensional, K-invariant state, the GCN output H^2 (Eq. (25)) is reduced to a graph-level readout $\mathrm{z}_{\mathrm{GCN}}(t)$ (Eq. (26)) and projected to $\mathrm{z}_{\mathrm{proj}}(t)$ (Eq. (27)). The state vector is therefore the concatenation of this projected graph readout and the task features:

$$S(t) = \left[\, \mathrm{z}_{\mathrm{proj}}(t);\, d_{r,i},\, c_{r,i},\, t_{r,i}^{\max} \,\right] \in \mathbb{R}^{18}, \tag{19}$$

where $\mathrm{z}_{\mathrm{proj}}(t) \in \mathbb{R}^{15}$ is the projected graph readout defined in Eqs. (26), (27), and the task features $d_{r,i}$, $c_{r,i}$, $t_{r,i}^{\max}$ complete the state. Because the mean-pooling readout is K-invariant, $\dim S(t) = 18$ is independent of the number of servers K. Here the task feature vector $v_{r,i} = \{d_{r,i}, c_{r,i}, t_{r,i}^{\max}\}$ implicitly encodes task execution progress through the remaining computational demand $c_{r,i}$, enabling the DRL agent to naturally avoid triggering unnecessary migrations for near-completion tasks. The 18-dimensional state $S(t)$ is composed of exactly two parts:

- a projected graph readout $z_{\text{proj}}(t) \in \mathbb{R}^{15}$, obtained by mean-pooling the GCN node embeddings $H^2 \in \mathbb{R}^{K \times F'}$ into a graph-level vector $z_{\text{GCN}}(t) \in \mathbb{R}^{F'}$ and applying a learned linear projection $W_p \in \mathbb{R}^{15 \times F'}$ (Eqs. (26), (27));
- the task features $[d_{r,i}, c_{r,i}, t_{r,i}^{\max}] \in \mathbb{R}^3$.

The resulting state dimension is 18 for any number of servers K. All input features are min-max normalized to [0, 1] using known physical deployment bounds to prevent feature-scale dominance and stabilize gradient computation.

Action Space: After the controller obtains the environmental state, the agent will make migration decisions based on the current state. This paper defines $\rho_{r,i}(t)$ as the migration decision of the vehicle when migration is triggered. When $\rho_{r,i}(t) = 1$, the proactive migration scheme is adopted, meaning the vehicle needs to migrate the incomplete task to the new MEC server for processing. When $\rho_{r,i}(t) = 0$, the passive migration scheme is adopted, meaning the current MEC server will not transfer the incomplete task but will wait until the task is completed and then return the results to the vehicle through the RSU communication link. Thus, the action space can be represented as:

$$A_{r,i}(t) = \rho_{r,i}(t) \tag{20}$$

Reward Function: The reward is defined as the negative of the normalized per-task cost, matching the per-task terms inside the objective of Eqs. (17)–(18):

$$R_{r,i}(t) = -\left(\varepsilon_1 \frac{t_{r,i}^{total}}{T_{\text{ref}}} + \varepsilon_2 \frac{e_{r,i}^{total}}{E_{\text{ref}}}\right). \tag{21}$$

Here, $t_{r,i}^{total}$ and $e_{r,i}^{total}$ are the per-task latency and energy resulting from action $\rho_{r,i}(t)$, and the same coefficients ε_1, ε_2 and the same per-task references T_{ref}, E_{ref} are used in both the optimization problem and the implemented reward.

Feature extraction

In the deep reinforcement learning framework, the agent located in the MEC controller needs to observe the state information of the environment and then process this information using a neural network. In vehicular networks, vehicle mobility changes server loads and link-state features over time, while the physical I2I adjacency remains fixed during normal operation. To capture the spatial dependencies between the time-varying node states on this graph, this paper uses a Graph Convolutional Network (GCN).

In the feature extraction process, this paper uses normalized node features such as computing resources $CPU(k)$, storage resources $Sto(k)$, and total bandwidth $Band(k)$ to represent the state of edge nodes. These features serve as the input for the first layer, which can be expressed as:

$$x_k = \left[CPU(k), Sto(k), Band(k)\right] \tag{22}$$

The adjacency matrix $A \in \mathbb{R}^{K \times K}$ is constructed from the physical I2I wired links between RSUs: $A_{kj} = 1$ if RSU k and RSU j are directly connected, and $A_{kj} = 0$ otherwise. Since RSU positions within a zone are fixed, A remains static during normal operation and is reconstructed only upon server failure or zone reconfiguration. This physical-connectivity definition is preferred over proximity-based or fully-connected alternatives, as it faithfully reflects the actual migration traffic graph while yielding a sparse structure compatible with the normalized GCN propagation described below.

The GCN uses the standard symmetrically normalized propagation operator with explicit self-loops. Let $\tilde{A} = A + I_K$ and let $\tilde{D}$ be its degree matrix, with $\tilde{D}_{kk} = \sum_j \tilde{A}_{kj}$. The layer-wise propagation rule is

$$H^{l+1} = \text{ReLU}\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^l W^l\right). \tag{23}$$

The self-loop term preserves each server's own CPU, storage, and bandwidth features at every layer, while symmetric degree normalization prevents node-dependent feature amplification on an irregular RSU graph. The physical adjacency A represents fixed I2I connectivity; dynamic link quality remains part of the time-varying node feature matrix $\mathrm{H}^0(t)$. The first and second GCN layers are therefore

$$\begin{aligned} \mathrm{H}^1 &= \mathrm{ReLU}\Big(\tilde{\mathrm{D}}^{-1/2}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{-1/2}\mathrm{H}^0\mathrm{W}^0\Big), \\ \tilde{\mathrm{A}} \in \mathbb{R}^{K\times K}, &\quad \mathrm{H}^0 \in \mathbb{R}^{K\times C}, \quad \mathrm{W}^0 \in \mathbb{R}^{C\times F}. \end{aligned} \tag{24}$$

$$\begin{aligned} \mathrm{H}^2 &= \mathrm{ReLU}\Big(\tilde{\mathrm{D}}^{-1/2}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{-1/2}\mathrm{H}^1\mathrm{W}^1\Big), \\ \mathrm{H}^1 \in \mathbb{R}^{K\times F}, &\quad \mathrm{W}^1 \in \mathbb{R}^{F\times F'}. \end{aligned} \tag{25}$$

The resulting row vector $\mathrm{h}_k(t)$ of $\mathrm{H}^2(t)$ is the topology-aware representation of server k. To obtain a fixed-dimensional, K-invariant graph-level representation from the per-server embeddings, a global mean-pooling readout is applied:

$$\mathrm{z}_{\mathrm{GCN}}(t) = \frac{1}{K}\sum_{k=1}^{K}\mathrm{h}_k(t) \in \mathbb{R}^{F'}, \tag{26}$$

where $\mathrm{h}_k(t)$ denotes the k-th row of $\mathrm{H}^2(t)$. Because the mean over nodes yields an F'-dimensional vector for any K, the readout is K-invariant. A learned linear projection then compresses this vector to a compact dimension d before it is fused with the task features:

$$\mathrm{z}_{\mathrm{proj}}(t) = \mathrm{W}_p\,\mathrm{z}_{\mathrm{GCN}}(t) \in \mathbb{R}^d, \tag{27}$$

with $\mathrm{W}_p \in \mathbb{R}^{d\times F'}$. Because $\mathrm{H}^0(t)$ contains time-varying resource and bandwidth states, the GCN is recomputed at every migration-decision epoch. Vehicle handoff $X_r = 1$ triggers a migration decision but does not by itself indicate that the infrastructure adjacency matrix has changed. A change in $\mathrm{A}(t)$ is represented by a separate infrastructure event and occurs only upon an RSU/server failure, recovery, or zone reconfiguration.

Migration decision algorithm

Input: Learning rates $\lambda_\theta, \lambda_w$; discount factor γ
Initialize actor parameters θ, state-value critic parameters w, and physical adjacency $\mathbf{A}(1)$
Set decay coefficient $I \leftarrow 1$
for $t \leftarrow 1$ to T **do**
    Retrieve the current task information $v_{r,i}(t)$ and MEC-server resource/link states
    Construct the current node feature matrix $\mathbf{H}^0(t)$
    if an infrastructure-topology event occurs **then**
        Update $\mathbf{A}(t)$ to reflect RSU/server failure, recovery, or zone reconfiguration
    else
        $\mathbf{A}(t) \leftarrow \mathbf{A}(t-1)$
    end if
    if $X_r = 1$ **then** ▷ vehicle handoff triggers a migration decision
        Recompute $\mathbf{H}^2(t) \leftarrow \mathrm{GCN}(\mathbf{A}(t), \mathbf{H}^0(t))$ and the readout $\mathbf{z}_{\mathrm{proj}}(t) \leftarrow \mathbf{W}_p\big(\frac{1}{K}\sum_{k=1}^{K}\mathbf{h}_k(t)\big)$
        Form $S(t) \leftarrow [\mathbf{z}_{\mathrm{proj}}(t)\,;\, v_{r,i}(t)]$ and select $a_t \sim \pi_\theta(\cdot \mid S(t))$
        Execute a_t, observe $R(t)$ and $S(t+1)$
        $\delta \leftarrow R(t) + \gamma\hat{v}(S(t+1), w) - \hat{v}(S(t), w)$
        $w \leftarrow w + \lambda_w I \delta \nabla_w \hat{v}(S(t), w)$
        $\theta \leftarrow \theta + \lambda_\theta I \delta \nabla_\theta \log \pi_\theta(a_t \mid S(t))$
        $I \leftarrow \gamma I$
    end if
    $S(t) \leftarrow S(t+1)$
end for

Algorithm 1 Migration decision algorithm process.

Given the frequent and random state changes due to vehicle mobility and urban road environments, this paper improves upon the traditional actor-critic algorithm. The improvement involves incorporating a Graph Convolutional Network (GCN) to extract features from edge nodes and link information, ensuring that each edge node incorporates information from neighboring nodes and links. This enhancement allows the model to generate more optimal task migration decisions. The specific algorithm model is illustrated in Fig. 2.

At each time slot, the MEC controller refreshes the node feature matrix $\mathrm{H}^0(t)$ from the current CPU, storage, and bandwidth states. Vehicle handoff $X_r = 1$ is treated solely as the trigger for a migration decision, whereas changes in physical RSU connectivity are represented by a separate infrastructure-topology event. At every migration-decision epoch, the controller recomputes $\mathrm{H}^2(t) = \mathrm{GCN}(\mathrm{A}(t), \mathrm{H}^0(t))$, applies the mean-pooling readout and the projection $\mathrm{z}_{\mathrm{proj}}(t) = \mathrm{W}_p \mathrm{z}_{\mathrm{GCN}}(t)$, and forms $S(t) = [\mathrm{z}_{\mathrm{proj}}(t); v_{r,i}(t)]$ before the actor selects $a_t \sim \pi_\theta(\cdot \mid S(t))$. Thus, no cached embedding is reused after node resources or link-state features have changed. The chosen action is executed in the environment, yielding a reward and the next state. Next, the critic network's update parameters, denoted as δ , are calculated based on $R(t) + \gamma\hat{v}(S(t+1), w) - \hat{v}(S(t), w)$, where $S(t)$ and $S(t+1)$ denote the current and next states, respectively, in accordance with the temporal-difference update in Algorithm 1. These update parameters δ serve as the guidance signal for the actor network, instructing it on how to adjust the action probabilities. The specific process is outlined in the pseudocode of Algorithm 1. The binary action space $\rho_{r,i}(t) \in \{0, 1\}$ is a deliberate design choice balancing decision granularity and algorithmic tractability. Expanding the action space to explicitly enumerate K candidate destination MEC servers increases policy gradient variance approximately proportional to $|\mathcal{A}|$, substantially degrading convergence stability under the non-stationary topology dynamics of IoV environments. Importantly, topology information remains incorporated in the decision process: the GCN aggregates the computing, storage, and bandwidth states across the K edge nodes into topology-aware node embeddings. These embeddings are subsequently transformed by the mean-pooling readout and linear projection into $z_{\mathrm{proj}}(t)$, which forms

part of the state $s_(t)$ used by the actor. Accordingly, the present method solves a deliberately restricted decision problem—the selection between active and passive migration schemes—while richer action spaces with explicit destination-level migration decisions are outside the present scope and are left for future work.

Gr-MiD adopts an on-policy Actor-Critic update, which renders replay buffers and target networks inapplicable by design, as these mechanisms are specific to off-policy value-based methods such as DQN. Exploration is handled intrinsically via the stochastic softmax policy: the actor outputs a probability distribution over the binary action space at each step, from which actions are sampled during training, providing continuous exploration that naturally anneals as policy entropy decreases. Each training episode comprises 5,000 discrete time slots and terminates upon exhausting all time slots without early stopping; training is concluded when the mean episodic reward stabilizes, typically within 800–1,000 episodes.

Performance evaluation

In this section, we first describe the simulation setup and then evaluate the performance of the proposed Gr-MiD algorithm.

Simulation setup

Experiment parameters

The simulation experiments were conducted on a desktop computer equipped with an Nvidia RTX 4090 GPU, using the Python NetworkX 3.1 library to generate the initial sparse physical RSU topology for each trial. The physical adjacency remains fixed during normal operation, while vehicle mobility produces time-varying server loads and link-state features. The considered vehicular network environment consists of one MEC controller, five MEC servers, and a set of vehicles. Each vehicle device is tasked with processing 10 tasks. It is assumed that the computational workload of each task is uniformly distributed between 10^3 and 10^4 mega-cycles (i.e., 10^9–10^{10} CPU cycles), while the MEC-server processing rate is measured in cycles/s. The data size for each task is uniformly distributed between $10 \sim 30$ MB. The maximum acceptable delay is between $0.5 \sim 1.2$ seconds. The computational capabilities of the vehicle devices and MEC servers are set to $2 \sim 5$ GHz and $15 \sim 35$ GHz, respectively[34]. Due to the fact that signal transmission power, channel gain, and noise power only affect the task transmission rate, this study directly sets the transmission rate R_{V2E} between vehicle devices and MEC servers to $40 \sim 60$ MB/s, and the transmission rate R_{E2E} between adjacent MEC servers to $100 \sim 150$ MB/s[35]. The link bandwidth is set to $1 \sim 10$ Gbps. Therefore, no specific settings are made for signal transmission power, channel gain, or noise power.

The Gaussian-Markov mobility model is widely adopted in MEC-oriented vehicular network research[23,26–28] for its ability to generate temporally correlated trajectories that reflect realistic movement inertia, inducing time-varying and spatially heterogeneous load distributions across MEC servers, which is the critical property for stress-testing migration decision policies. While datasets such as T-Drive[36] offer high-fidelity trajectories, they lack the application-layer workload parameters (e.g., per-task computational demand and data volume) required by MEC simulators, making direct integration non-trivial. Regarding transmission rates, the fixed ranges (V2E: 40–60 MB/s; E2E: 100–150 MB/s) represent scenario-averaged throughputs consistent with the literature[34,35] and reflect a deliberate system-level abstraction, as the focus of this work is the migration decision policy rather than physical-layer phenomena such as fading or interference. Incorporating stochastic channel models and realistic mobility traces are identified as directions for future work.

The experimental configuration of $K = 5$ MEC servers corresponds to a single MEC controller zone, which is the standard intra-zone operational unit of the proposed framework and is consistent with the deployment scales adopted in representative IoV-MEC studies[26–28]. Large-scale deployments are realized by replicating this zone structure under multiple MEC controllers. Since A encodes a sparse road-following topology with $|\mathcal{E}| = \mathcal{O}(K)$

edges, the per-layer GCN complexity reduces to $\mathcal{O}(K \cdot F)$. Moreover, the mean-pooling readout and the linear projection are K-invariant: $H^2 \in \mathbb{R}^{K \times F'}$ is mapped to a fixed 15-dimensional vector $z_{proj}(t)$ for any K, so the actor–critic input dimension remains 18 regardless of the number of servers. The framework therefore scales linearly in computation with zone size and accepts variable graph sizes without architectural modification.

Statistical methodology

The study uses synthetic data generated with Python NetworkX to simulate IoV-MEC environments, ensuring all algorithms are evaluated on identical datasets. All algorithms (Gr-MiD, DQN, DDQN, DDPG) process the same vehicle mobility traces and task arrival sequences, with results averaged over 30 independent trials to mitigate randomness. Unless otherwise stated, performance curves report mean values with error bars representing 95% confidence intervals across the 30 trials. Figure 3 reports the median normalized reward; its 95% confidence interval is obtained by non-parametric bootstrap resampling (10,000 resamples) over the 30 independent trials. Each trial is governed by a fixed random seed (seeds 0–29) applied consistently across all compared algorithms, covering mobility trace generation, topology initialization, and neural network weight initialization, ensuring full reproducibility.

To verify real-time feasibility, we profiled the end-to-end decision latency over 500 consecutive runs on a workstation equipped with an NVIDIA GeForce RTX 4090 GPU. The GCN inference averaged 0.8 ms, and the complete pipeline (GCN feature extraction and actor forward pass) averaged 2.1 ms, which is less than 1.1% of the 200 ms decision period $\Delta t = 0.2$ s (5 decisions/s). These results indicate that the proposed framework incurs a relatively small decision-time overhead on the evaluated GPU platform.

The structure of neural network

According to the state-space definition in Eq. 19, $S(t)$ consists of the 15-dimensional projected graph readout $z_{proj}(t)$ (Eqs. (26), (27)) and the 3-dimensional task features, giving 18 scalar features in total.The actor therefore uses an 18-dimensional input and produces a two-dimensional probability distribution over the active/passive actions. Algorithm 1 employs a state-value critic $\hat{v}(S(t), w)$; accordingly, the critic receives only the current 18-dimensional state and outputs one scalar value. It does not concatenate the action, next state, or reward with its

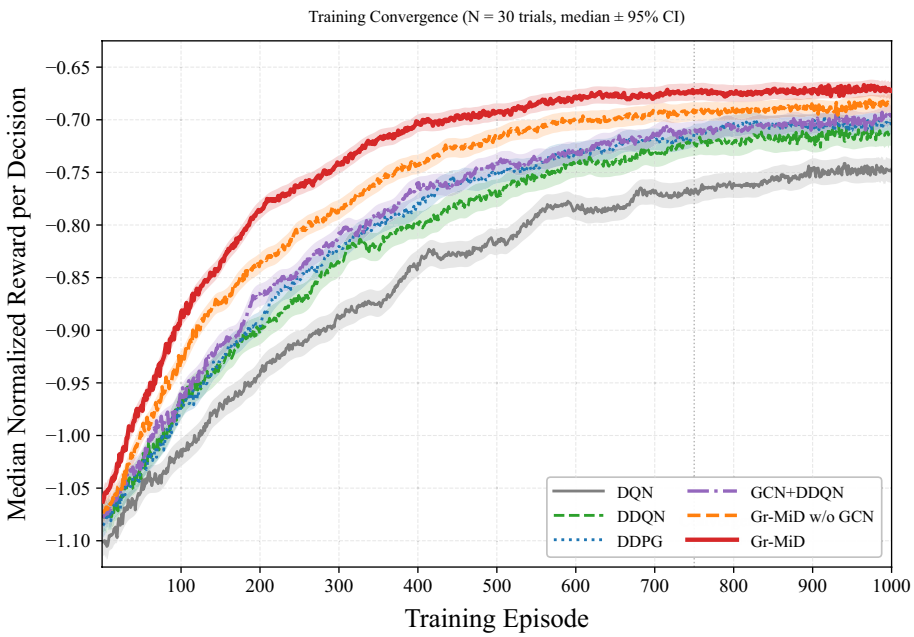

Fig. 3 Convergence curves of all compared algorithms over 1,000 training episodes (median ± 95% CI across 30 independent trials, smoothed with a 50-episode sliding window). The 95% CI for the median is obtained by non-parametric bootstrap resampling (10,000 resamples). Statistical summaries are computed over the final 100 episodes. Gr-MiD achieves the highest converged reward and most stable training trajectory among all methods.

input. The actor and critic architectures are therefore $18 \times 256 \times 256 \times 256 \times 2$ and $18 \times 256 \times 256 \times 256 \times 1$, respectively. The hidden-layer width is selected empirically, and the GCN, together with the mean-pooling readout and the linear projection, provides the topology-aware state representation used by the actor–critic model. The simulation environment is built using Python, and TensorFlow is used for constructing and training the neural networks. The learning rates for the actor and critic networks are set to 0.001 and 0.005, respectively, with a discount factor set to 0.9. All parameter information is summarized in Table 2.

Performance comparison

Comparison of different algorithms

This section compares Gr-MiD with five comparison methods, including three conventional DRL baselines (DQN, DDQN, and DDPG), one graph-based baseline (GCN+DDQN), and one ablation variant (Gr-MiD w/o GCN).

- DQN[27]: The DQN algorithm is a method that combines Q-learning with neural networks, using the latter to learn the Q-value function. In[27], the authors proposed a task migration decision algorithm based on DQN to effectively reduce network service delay and migration delay.
- DDQN[26]: The DDQN algorithm is an improved deep reinforcement learning algorithm that uses dual value networks to mitigate overestimation issues. In[26], the authors applied DDQN to address task migration decision problems in vehicular networks, aiming to minimize total service delay and migration costs.
- DDPG[28]: The DDPG algorithm is a method that combines deep learning with policy gradient techniques. In[28], the authors used DDPG to make rapid migration decisions, focusing on minimizing migration costs related to delay and energy consumption.
- Gr-MiD: The proposed GCN-based deep reinforcement learning algorithm in this paper aims to minimize the delay and energy consumption of vehicular task migration in vehicular networks.
- GCN+DDQN[31]: GCN+DDQN integrates Graph Convolutional Networks with Double Deep Q-Networks for topology-aware offloading decisions in vehicular edge computing. This baseline is included to disentangle the contribution of graph-based representations from that of the Actor-Critic framework adopted in Gr-MiD.

Table 2 Model training parameters.

Parameter category	Parameter name	Value
Neural network architecture	Actor network input dimensions	18 (15 + 3)
	Actor network hidden layers	3 × 256
	Actor network output dimensions	2
	Critic network input dimensions	18 (15 + 3)
	Critic network hidden layers	3 × 256
	Critic network output dimensions	1
GCN configuration	GCN layers	2
	Initial node feature dimensions	3
	Convolutional kernel dimensions	256 → 256
	Readout Projection dimension (d)	15
Training hyperparameters	Learning Rate (Actor)	0.001
	Learning Rate (Critic)	0.005
	Discount Factor (γ)	0.9
Environment parameters	Number of MEC servers	5
	Vehicle count range	10–40
	Tasks per vehicle	10
	Scheduling interval (Δt)	0.2 s
	Per-task Latency Reference (T_{ref})	1.5 s
	Per-task energy reference (E_{ref})	60 J
	MEC server CPU range	15–35 GHz

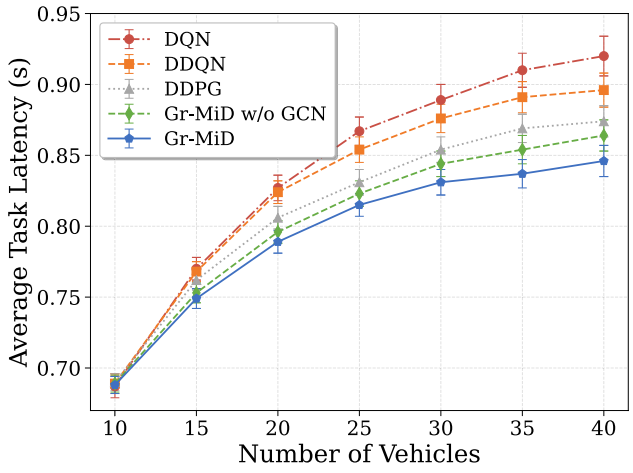

Fig. 4 Comparison of average task latency with different algorithms and ablation baseline (Gr-MiD w/o GCN).

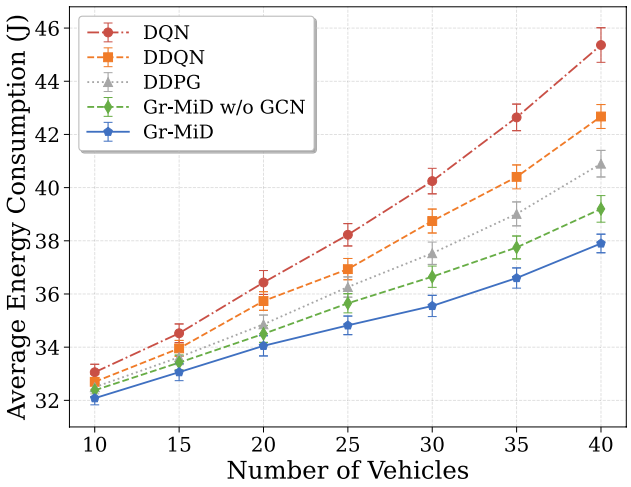

Fig. 5 Comparison of average task energy with different algorithms and ablation baseline (Gr-MiD w/o GCN).

- Gr-MiD w/o GCN: Gr-MiD w/o GCN is an ablation variant of the proposed algorithm in which the GCN feature extractor is replaced by a parameter-equivalent MLP that processes the raw concatenated node feature vector directly, with all other components held constant, in order to isolate the contribution of graph-structured neighborhood aggregation.

Additionally, the latency and energy comparisons in Figs. 4, 5, 6 use paired t-tests with Bonferroni correction for the three pre-specified baseline comparisons ($k = 3$). The converged-reward comparisons in Fig. 3 use Wilcoxon signed-rank tests with correction for five comparisons ($k = 5$). Effect sizes are reported using Cohen's d to assess practical significance.

Figure 3 presents the training convergence curves of all compared algorithms over 1,000 episodes (30 independent trials, smoothed with a 50-episode sliding window), with statistical summaries computed over the final 100 episodes. Defining the positive normalized cost as $C = -R$, Gr-MiD reduces the converged normalized cost relative to DQN, DDQN, DDPG, GCN+DDQN[31], and Gr-MiD w/o GCN by 10.25%, 5.91%, 4.70%, 4.18%, and 2.31% , respectively (all Bonferroni-adjusted $p < 0.05$, Wilcoxon signed-rank test; $d = 2.78, 1.50, 1.02, 0.58, 0.46$). The larger effect sizes over DQN, DDQN, and DDPG support the advantage of on-policy Actor-Critic updates under non-stationary migration dynamics. Gr-MiD w/o GCN converges to a higher reward than GCN+DDQN, while the complete Gr-MiD model achieves the best performance. This pattern indicates that both the Actor-Critic update and the GCN representation contribute to the final result under the evaluated setting.

Figure 4 compares the average task delay of different algorithms as the number of vehicles changes. As illustrated, with an increase in the number of vehicles, the available communication and computing resources on each MEC server decrease, leading to an increase in average delay. When the number of vehicles is low, MEC servers can allocate sufficient computing resources to each vehicle device, resulting in minimal performance differences among the various methods. However, as the number of vehicles increases and task migration decisions become more critical, the proposed Gr-MiD method, which comprehensively considers bandwidth resources and the state of neighboring MEC servers, can better address load balancing among MEC servers. Consequently, Gr-MiD achieves improved performance. Specifically, when the number of vehicles reaches 40, the Gr-MiD method provides a delay performance improvement of 8.04%, 5.58%, and 3.20% compared to the DQN, DDQN, and DDPG algorithms, respectively. Paired t-tests across 30 simulation runs confirm statistical significance: Gr-MiD vs.

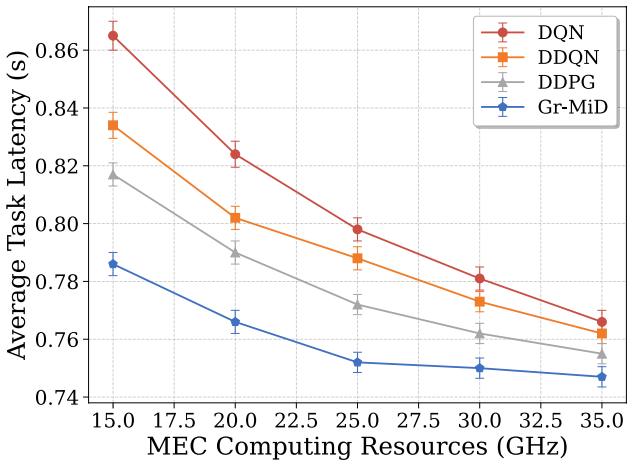

Fig. 6 Impact of MEC server computing resources on average latency of tasks with different algorithms.

DQN ($p < 0.001$), vs. DDQN ($p = 0.003$), and vs. DDPG ($p = 0.012$), all remaining significant after Bonferroni correction. The ablation baseline Gr-MiD w/o GCN, which replaces the GCN module with a parameter-equivalent MLP while keeping the Actor-Critic framework intact, consistently falls between Gr-MiD and DDPG. At 10 vehicles, its latency is nearly identical to Gr-MiD (0.6890 s vs. 0.6880 s), indicating that GCN provides marginal gain under light load. As the vehicle count grows to 40, the gap between Gr-MiD and Gr-MiD w/o GCN widens to 2.08%, while Gr-MiD w/o GCN still outperforms DDPG by 1.14%, isolating the respective contributions of the Actor-Critic framework and the GCN topology encoder to the overall performance gain.

Figure 5 compares the average task energy consumption of different algorithms as the number of vehicles varies. As shown, with an increase in the number of vehicles, the Gr-MiD strategy demonstrates better performance. This is attributed to the algorithm's incorporation of GCN for network topology feature extraction, allowing it to better perceive the network environment. Similar to Fig. 4, the average task energy consumption increases with the number of vehicles because more tasks need to be migrated as the vehicle count grows, resulting in higher migration energy consumption. When the number of vehicles is low, the differences among the three algorithms are not significant. However, as the number of vehicles increases, the Gr-MiD strategy exhibits superior performance. Specifically, with 40 vehicles, the Gr-MiD strategy reduces the average task energy consumption by approximately 15.82%, 10.09%, and 6.36% compared to the DQN, DDQN, and DDPG strategies, respectively. Statistical validation through paired t-tests confirms these energy reductions are significant: Gr-MiD vs. DQN ($p < 0.001$), vs. DDQN ($p < 0.001$), and vs. DDPG ($p = 0.005$), with all comparisons remaining significant after Bonferroni correction. A consistent pattern is observed for energy consumption. The gap between Gr-MiD and Gr-MiD w/o GCN grows from negligible at 10 vehicles to 2.79% at 40 vehicles, while Gr-MiD w/o GCN reduces energy consumption by 3.67% compared to DDPG at the same load. This monotonically increasing divergence confirms that the GCN's multi-hop neighborhood aggregation becomes increasingly critical for energy-efficient migration decisions as network topology complexity grows, and that the Actor-Critic framework itself contributes an independent, non-trivial performance gain over value-based methods.

Figure 6 illustrates the impact of MEC server computing resources on the average task latency of different migration decision algorithms. It is important to note that in this experiment, the number of vehicles is fixed at 20. As shown in Fig. 6, the average task latency for all strategies decreases with the increase in MEC server computing resources. This trend occurs because enhanced computing capabilities of MEC servers reduce the task processing delay. When the computing power of MEC servers is limited, the Gr-MiD strategy can reduce task latency by making appropriate migration decisions, thanks to its comprehensive perception of the entire edge network. As the computing resources of edge servers improve, the differences between the strategies gradually diminish because the impact of migration strategies on the overall system latency becomes smaller. Specifically, when the MEC server computing resources are 15 GHz, the Gr-MiD strategy provides approximately 9.13%,

5.76%, and 3.79% performance improvement in comparison to the DQN, DDQN, and DDPG strategies, respectively. Statistical validation confirms these improvements are significant: Gr-MiD vs. DQN ($p < 0.001$), vs. DDQN ($p = 0.002$), and vs. DDPG ($p = 0.008$), with all comparisons remaining significant after correction.

Figure 7 reports the latency–energy trade-off for $\varepsilon_1 \in \{0.01, 0.10, 0.20, \ldots, 0.90, 0.99\}$ with $\varepsilon_2 = 1 - \varepsilon_1$. For each weight pair, a separate Gr-MiD policy is trained from scratch using the same 1,000-episode protocol, random seeds, network architecture, and hyperparameters; the plotted latency and energy are evaluated using the converged policy corresponding to that weight pair. The weights are not changed only at inference time. The number of vehicles is fixed at 30 and each point is averaged over 30 independent trials. As ε_1 increases, the learned policy assigns greater importance to normalized latency and correspondingly less importance to normalized energy. This produces the monotonic latency–energy trade-off shown in the figure and confirms that the normalized reward is well-conditioned across the tested preference range. The balanced operating point near $\varepsilon_1 = 0.5$ is used as the default configuration in the remaining experiments.

While the delay gain of Gr-MiD over DDPG is 3.20% at 40 vehicles, this is a comparatively modest improvement under the default experimental setting. As shown in Fig. 6, when MEC computing resources are constrained to 15 GHz, the improvement over DDPG increases to 3.79%. In the corresponding energy evaluation at 40 vehicles, Gr-MiD achieves a 6.36% reduction over DDPG (Fig. 5). In a separate migration-count evaluation, the Gr-MiD-based IMD scheme reduces migration counts by 36.89% compared with FAM (Fig. 10). The GCN inference latency averages 0.8 ms, while the complete decision pipeline averages 2.1 ms, corresponding to approximately 1.05% of the 200 ms decision window and confirming a favorable performance–overhead trade-off.

Performance comparison between different migration schemes

To validate the effectiveness of the proposed migration method, this study compares four schemes: three comparison heuristics and one the proposed method, they are described as follows:

- Full Active Migration (FAM) : In this scheme, service instances on the MEC servers will always migrate to the MEC server closest to the vehicle device to provide computing services.
- Full Passive Migration (FPM) : In this scheme, service instances on the MEC servers will always remain on the original MEC server without proactive migration, until the latency fails to meet the requirements.
- Partial Active Migration (PAM): In this scheme, service instances on the MEC servers will randomly adopt the active migration scheme with a probability of 50%.

- Intelligent Migration Decision (IMD): it is our proposed method, in this scheme, the Gr-MiD model on the MEC controller makes comprehensive migration decisions based on the features of edge nodes, communication

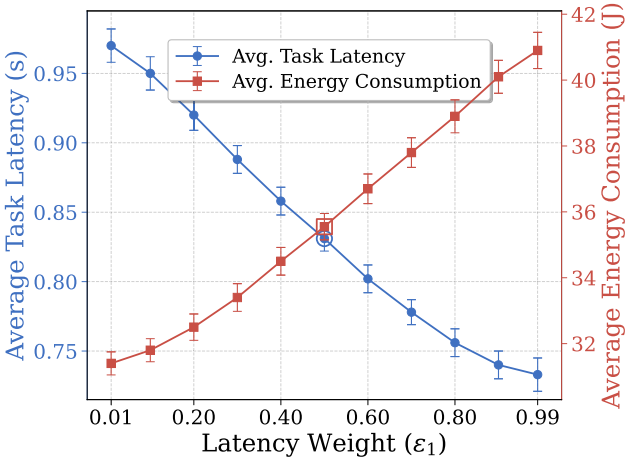

Fig. 7 Sensitivity analysis of latency-energy trade-off in Gr-MiD algorithm.

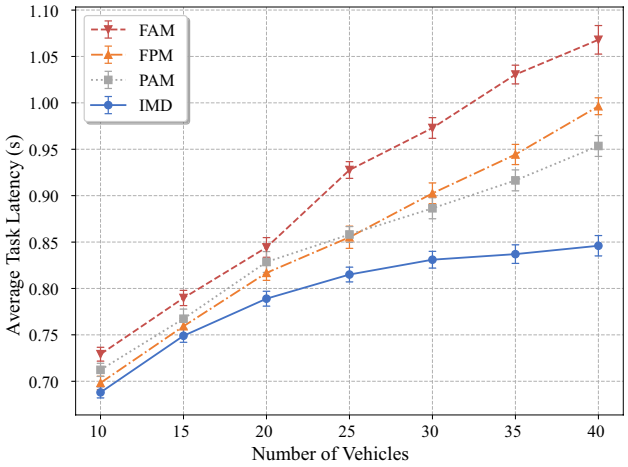

Fig. 8 Comparison of the average latency of tasks in different migration schemes.

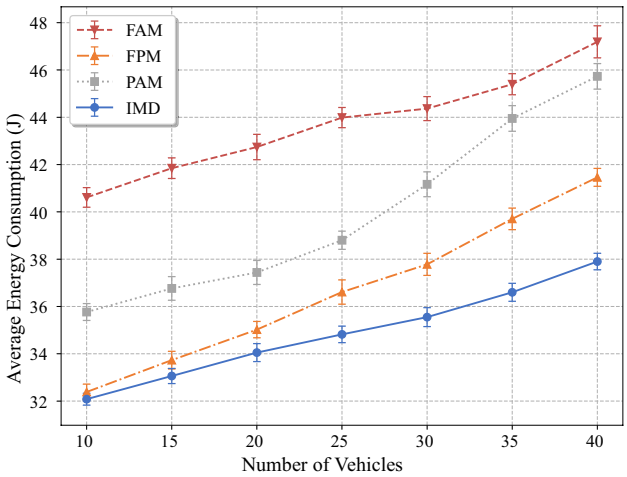

Fig. 9 Comparison of the average task energy consumption under different migration schemes.

links, and vehicle tasks.*Rationality of Synthetic Data:* The data distribution is based on typical scenario assumptions for edge computing in IoV ([19,26,27]). The parameter ranges refer to real - vehicle tasks (e.g., data processing requirements for autonomous driving sensors, as detailed in[1,3]).

- *Dynamic Simulation:* Vehicle movement trajectories are randomly generated using the Gaussian–Markov model to trigger handovers between RSU coverage areas and dynamic changes in the server-load and link-state features; the physical RSU adjacency changes only under explicit infrastructure events. Each simulation includes 5000 time steps. Within each time step, vehicles may move to the coverage of different RSUs, triggering migration decisions.

The average task delay of different migration schemes as a function of the number of vehicles is shown in Fig. 8. As the number of vehicles increases, the average task delay for all migration schemes also increases. This is due to the increased number of tasks that the MEC servers need to handle as the number of vehicles grows, leading to increased queuing delays. When the number of vehicles is low, the FPM scheme consistently has a lower delay compared to the FAM and PAM schemes. This is because the MEC server resources are sufficient and the number of tasks to be migrated is small, so there is no migration delay in the FPM scheme. However, as the number of vehicles increases, the FPM scheme faces issues such as MEC server overload and load imbalance, causing its average task delay to eventually exceed that of the PAM scheme. According to the results, when the number of vehicles reaches 40, the proposed IMD scheme achieves approximately 20.78%, 11.28%, and 15.09% performance improvements compared to the FAM, PAM, and FPM schemes, respectively.

Figure 9 compares the energy consumption of different migration schemes as a function of the number of vehicles. As shown in the figure, the average energy consumption of tasks increases with the number of vehicles. This is because, with the increase in vehicle numbers, the load on MEC servers intensifies, leading to a greater number of tasks that need to be migrated, resulting in higher migration or transmission energy consumption. When the number of vehicles is low, the performance of the FPM strategy and the IMD strategy is similar. This is because, with fewer vehicles, MEC servers have sufficient computational resources and the number of tasks needing migration is small. Consequently, FPM incurs no migration energy costs, leading to minimal differences in energy consumption. However, as the number of vehicles increases, the advantages of the IMD scheme become more pronounced. Specifically, when the number of vehicles reaches 40, the IMD scheme achieves

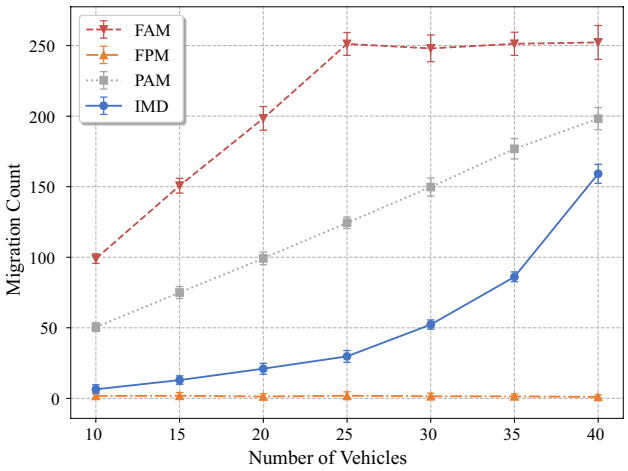

Fig. 10 Comparison of the number of task migrations under different migration schemes.

approximately 19.69%, 17.13%, and 8.59% performance improvements compared to the FAM, PAM, and FPM schemes, respectively.

Figure 10 shows the comparison of task migration counts under different numbers of vehicles. As depicted, the task migration counts for the FAM, PAM, and IMD schemes all increase with the number of vehicles. Notably, for the FAM scheme, the task migration count saturates near the preset maximum threshold when the number of vehicles reaches 25. This is due to the communication resource constraints of the MEC servers, where a maximum migration threshold was set in this simulation experiment for the FAM scheme. As the number of vehicles increases, the IMD scheme shows a greater advantage. Specifically, when the number of vehicles reaches 40, the IMD scheme reduces the migration count by 36.89% compared to the FAM scheme and by 19.70% compared to the PAM scheme.

Conclusion

In this paper, we investigate the vehicle task migration decision problem in the dynamic IoV-MEC scenario, which is characterized by real-time changes in vehicle locations, server loads, link-state features, and resource environments over a physical RSU graph. Specifically, we first model the vehicle task migration decision problem in the IoV-MEC network as a multi-objective optimization problem, aiming to minimize the migration cost, which is composed of task migration delay and energy consumption. To address the challenge of time-varying network states and resource environments in vehicular networks, we use a Graph Convolutional Network (GCN) to extract features of network topology and edge node resources. Then, we employ Deep Reinforcement Learning (DRL) to make task migration decisions, considering both network and task features, in order to optimize task migration delay and energy consumption. The simulation results demonstrate that the proposed Gr-MiD algorithm significantly outperforms existing baseline algorithms in the evaluated small synthetic intra-zone IoV-MEC setting. Specifically, under peak-load conditions (40 vehicles, averaged over 30 independent trials), Gr-MiD achieves an 8.04% reduction in average task delay and a 15.82% reduction in average task energy consumption compared to the DQN baseline. Furthermore, the intelligent migration decision scheme achieves a 20.78% reduction in average task latency and reduces migration counts by 36.89% compared to conventional full active migration.

The current evaluation also has three limitations that motivate these extensions: empirical validation is confined to a single zone of K=5 servers; mobility traces are synthetically generated rather than drawn from real-world datasets; and radio resource parameters are treated as fixed environmental constants.

Future work will explore graph attention mechanisms (e.g., GAT) to enable adaptive neighbor weighting under highly dynamic link-quality conditions. Additionally, three further extensions are planned: refining the migration cost model with dynamic container image size estimation; generalizing the framework to multi-RAT environments where vehicles access RSUs via heterogeneous technologies such as DSRC, C-V2X, and 5G NR; and exploring integration with 6G AI-RAN architectures to enable AI-native cross-layer migration orchestration.

Author contributions L.W. and L.L. conceived and designed this paper; W.L. and H.F. performed the experiments and analyzed the data; L.W. and X.Z. wrote the paper; L.L. and H.L. provided the funding support. All authors reviewed the manuscript.

Funding This research was funded by the National Natural Science Foundation of China (No. 62171070, 62402298), the Key Project of Chongqing Education and Science "14th Five-Year Plan" in 2023 (No. K23YD2060091), the Fundamental Research Program of Shanxi Province (No. 202403021212167).

Data availability All data generated or analysed during this study are included in this published article.

Declarations

Competing interests The authors declare no competing interests.

References

1. Chib, P. S. & Singh, P. Recent advancements in end-to-end autonomous driving using deep learning: A survey. *IEEE Trans. Intell. Veh.* **9**(1), 103–118. https://doi.org/10.1109/TIV.2023.3318070 (2023).
2. Zhao, J. et al. Autonomous driving system: A comprehensive survey. *Expert Syst. Appl.* **242**, 122836. https://doi.org/10.1016/j.eswa.2023.122836 (2024).
3. Mun, H. et al. Privacy enhanced data aggregation based on federated learning in internet of vehicles (iov). *Computer Commun.* **223**, 15–25. https://doi.org/10.1016/j.comcom.2024.05.009 (2024).
4. Sadatdiynov, K. et al. A review of optimization methods for computation offloading in edge computing networks. *Digital Commun. Netw.* **9**(2), 450–461. https://doi.org/10.1016/j.dcan.2022.03.003 (2023).
5. Mishra, K., Rajareddy, G. N., Ghugar, U., Chhabra, G. S. & Gandomi, A. H. A collaborative computation and offloading for compute-intensive and latency-sensitive dependency-aware tasks in dew-enabled vehicular fog computing: A federated deep q-learning approach. *IEEE Trans. Netw. Service Manag.* **20**(4), 4600–4614. https://doi.org/10.1109/TNSM.2023.3282795 (2023).
6. Trabelsi, Z., Ali, M. & Qayyum, T. Fuzzy-based task offloading in internet of vehicles (iov) edge computing for latency-sensitive applications. *Internet Things* **28**, 101392. https://doi.org/10.1016/j.iot.2024.101392 (2024).
7. Liang, C. et al. Ealso: joint energy-aware and latency-sensitive task offloading for artificial intelligence of things in vehicular fog computing. *Wirel. Netw.* 1–17 (2024).
8. Deng, T., Chen, Y., Chen, G., Yang, M. & Du, L. Task offloading based on edge collaboration in mec-enabled iov networks. *J. Commun. Netw.* **25**(2), 197–207. https://doi.org/10.23919/JCN.2023.000004 (2023).
9. Moghaddasi, K., Rajabi, S. & Gharehchopogh, F. S. Multi-objective secure task offloading strategy for blockchain-enabled iov-mec systems: a double deep q-network approach. *IEEE Access* **12**, 3437–3463. https://doi.org/10.1109/ACCESS.2023.3348513 (2024).

10. Chen, R., Fan, Y., Yuan, S. & Hao, Y. Vehicle collaborative partial offloading strategy in vehicular edge computing. *Mathematics* **12**(10), 1466. https://doi.org/10.3390/math12101466 (2024).
11. Miao, Y. et al. Intelligent task prediction and computation offloading based on mobile-edge cloud computing. *Future Gener. Comput. Syst.* **102**, 925–931. https://doi.org/10.1016/j.future.2019.09.035 (2020).
12. Zhang, X., Wu, W., Wang, J. & Liu, S. Bilstm-based federated learning computation offloading and resource allocation algorithm in mec. *ACM Trans. Sens. Netw.* **19**(3), 1–20. https://doi.org/10.1145/3579824 (2023).
13. Huang, S., Wen, J. & Chen, Z. Intelligent service migration towards mec-based iov systems. *J. Syst. Simul.* **37**(2), 379 (2025).
14. Abkenar, F. S. et al. A survey on mobility of edge computing networks in iot: State-of-the-art, architectures, and challenges. *IEEE Commun. Surveys Tutor.* **24**(4), 2329–2365. https://doi.org/10.1109/COMST.2022.3211462 (2022).
15. Wang, X., Wang, S., Gao, X., Qian, Z. & Han, Z. Amtos: An admm-based multi-layer computation offloading and resource allocation optimization scheme in iov-mec system. *IEEE Internet Things J.* (2024).
16. Labriji, I. et al. Mobility aware and dynamic migration of mec services for the internet of vehicles. *IEEE Trans. Netw. Service Manag.* **18**(1), 570–584. https://doi.org/10.1109/TNSM.2021.3052808 (2021).
17. Sun, G. et al. Profit maximization of independent task offloading in mec-enabled 5g internet of vehicles. *IEEE Trans. Intell. Transport. Syst.* https://doi.org/10.1109/TITS.2024.3416300 (2024).
18. Chen, Z. et al. Mobility-aware seamless service migration and resource allocation in multi-edge iov systems. *IEEE Trans. Mobile Comput.* (2025).
19. Kim, T. et al. Modems: Optimizing edge computing migrations for user mobility. *IEEE J. Select. Areas Commun.* **41**(3), 675–689. https://doi.org/10.1109/JSAC.2022.3229425 (2022).
20. Liu, Z. & Xu, X. Latency-aware service migration with decision theory for internet of vehicles in mobile edge computing. *Wirel. Netw.* **30**(5), 4261–4273 (2024).
21. Abouaomar, A., Mlika, Z., Filali, A., Cherkaoui, S. & Kobbane, A. A deep reinforcement learning approach for service migration in mec-enabled vehicular networks. In *2021 IEEE 46th Conference on Local Computer Networks (LCN)* 273–280. https://doi.org/10.1109/LCN52139.2021.9524882 (2021).
22. Zhou, A., Wang, S., Ma, X. & Yau, S. S. Towards service composition aware virtual machine migration approach in the cloud. *IEEE Trans. Services Comput.* **13**(4), 735–744. https://doi.org/10.1109/TSC.2019.2962128 (2019).
23. Wang, D., Tian, X., Cui, H. & Liu, Z. Reinforcement learning-based joint task offloading and migration schemes optimization in mobility-aware mec network. *China Commun.* **17**(8), 31–44. https://doi.org/10.23919/JCC.2020.08.003 (2020).
24. Li, J., Liang, W., Chen, M. & Xu, Z. Mobility-aware dynamic service placement in d2d-assisted mec environments. In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)* 1–6. https://doi.org/10.1109/WCNC49053.2021.9417358 (IEEE, 2021).
25. Wang, H., Lv, T., Lin, Z. & Zeng, J. Energy-delay minimization of task migration based on game theory in mec-assisted vehicular networks. *IEEE Trans. Veh. Technol.* **71**(8), 8175–8188. https://doi.org/10.1109/TVT.2022.3175238 (2022).
26. Chen, J., Xing, H., Xiao, Z., Xu, L. & Tao, T. A drl agent for jointly optimizing computation offloading and resource allocation in mec. *IEEE Internet Things J.* **8**(24), 17508–17524. https://doi.org/10.1109/JIOT.2021.3081694 (2021).
27. Zhang, Y., Li, R., Zhao, Y. & Li, R. Deep reinforcement learning based mobility-aware service migration for multi-access edge computing environment. In: *2022 IEEE Symposium on Computers and Communications (ISCC)*, 1–6. https://doi.org/10.1109/ISCC55528.2022.9912842. (IEEE, 2022).
28. Luo, Q., Zhang, J., Hu, S., Luan, T. H. & Fan, P. Joint task migration and resource allocation in vehicular edge computing: A deep reinforcement learning-based approach. *IEEE Trans. Veh. Technol.* (2025).
29. Zhao, L. et al. T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Trans. Intell. Transport. Syst.* **21**(9), 3848–3858. https://doi.org/10.1109/TITS.2019.2935152 (2020).
30. Xu, Y., Lu, Y., Dai, Y., Sun, C., Liu, J., Zhang, W. & Wu, H. Federated graph neural networks for dynamic computation offloading in vehicular networks. In: *GLOBECOM 2024 - 2024 IEEE Global Communications Conference*, 4196–4201. https://doi.org/10.1109/GLOBECOM52923.2024.10901545 (2024).
31. Ullah, I. & Han, Y.-H. Optimizing vehicular edge computing: graph-based double-dqn approaches for intelligent task offloading. *J. Supercomput.* **81**(1), 76 (2025).
32. Luo, F., Luo, C., Wang, J., Li, Z., Liao, Z. & Liu, Q. Anomaly detection in vehicular networks using causality-aware graph convolutional networks (ca-gcn). *Int. J.Automot. Technol.*, 1–16 (2025)
33. Zhang, Q. et al. A comparative study of containers and virtual machines in big data environment. In *IEEE 11th International Conference on Cloud Computing (CLOUD)* 178–185 (IEEE, 2018).
34. Wu, H., Zhang, Z., Guan, C., Wolter, K. & Xu, M. Collaborate edge and cloud computing with distributed deep learning for smart city internet of things. *IEEE Internet Things J.* **7**(9), 8099–8110. https://doi.org/10.1109/JIOT.2020.2996784 (2020).
35. Lekharu, A., Chauhan, A. P. S., Sur, A. & Patra, M. Reinforcement learning based adaptive bitrate caching at mec server. *IEEE Trans. Netw. Service Manag.* https://doi.org/10.1109/TNSM.2024.3367333 (2024).

36. Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., Sun, G., & Huang, Y. T-drive: driving directions based on taxi trajectories. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 99–108 (2010)

Authors and Affiliations

Liyan Wang[1,2] · Xianfeng Zheng[1,2] · Liang Liu[3] · Hao Feng[4] · Wenwei Li[3] · Huan Liu[5]

✉ Liang Liu
liuliang@cqupt.edu.cn

✉ Huan Liu
xq_liuhuan@tmmu.com.cn

1 School of Computing, Chongqing College of Mobile Communication, Jiari Road, Hechuan, Chongqing 401520, China

2 Chongqing Key Laboratory of Public Big Data Security Technology, Chongqing College of Mobile Communication, Tonghui, Qijiang, Chongqing 401420, China

3 School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongwen Road, Chongqing 400065, Chongqing, China

4 School of Information, Shanxi University of Finance and Economics, Wucheng Road, Taiyuan 030006, Shanxi, China

5 Department of Orthopedics, The Second Affiliated Hospital of Army Medical University, Xinqiao Zheng Street, Chongqing 400037, Chongqing, China