Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.068795

ARTICLE

A Multi-Objective Deep Reinforcement Learning Algorithm for Computation Offloading in Internet of Vehicles

Junjun Ren[1], **Guoqiang Chen**[2], **Zheng-Yi Chai**[3] **and Dong Yuan**[4,*]

[1]School of Computer and Big Data Engineering, Zhengzhou Business University, Zhengzhou, 451400, China
[2]School of Computer and Information Engineering, Henan University, Kaifeng, 475000, China
[3]School of Joint Innovation Industry, Quanzhou Vocational and Technical University, Quanzhou, 362268, China
[4]School of Computer Science and Technology, Xidian University, Xi'an, 710071, China

*Corresponding Author: Dong Yuan. Email: yd@stu.xidian.edu.cn

Received: 06 June 2025; Accepted: 06 June 2025; Published: 10 November 2025

ABSTRACT: Vehicle Edge Computing (VEC) and Cloud Computing (CC) significantly enhance the processing efficiency of delay-sensitive and computation-intensive applications by offloading compute-intensive tasks from resource-constrained onboard devices to nearby Roadside Unit (RSU), thereby achieving lower delay and energy consumption. However, due to the limited storage capacity and energy budget of RSUs, it is challenging to meet the demands of the highly dynamic Internet of Vehicles (IoV) environment. Therefore, determining reasonable service caching and computation offloading strategies is crucial. To address this, this paper proposes a joint service caching scheme for cloud-edge collaborative IoV computation offloading. By modeling the dynamic optimization problem using Markov Decision Processes (MDP), the scheme jointly optimizes task delay, energy consumption, load balancing, and privacy entropy to achieve better quality of service. Additionally, a dynamic adaptive multi-objective deep reinforcement learning algorithm is proposed. Each Double Deep Q-Network (DDQN) agent obtains rewards for different objectives based on distinct reward functions and dynamically updates the objective weights by learning the value changes between objectives using Radial Basis Function Networks (RBFN), thereby efficiently approximating the Pareto-optimal decisions for multiple objectives. Extensive experiments demonstrate that the proposed algorithm can better coordinate the three-tier computing resources of cloud, edge, and vehicles. Compared to existing algorithms, the proposed method reduces task delay and energy consumption by 10.64% and 5.1%, respectively.

KEYWORDS: Deep reinforcement learning; internet of vehicles; multi-objective optimization; cloud-edge computing; computation offloading; service caching

1 Introduction

The Internet of Vehicles (IoV) extends the Internet of Things (IoT) paradigm into the vehicular domain [1], and has great potential in enabling intelligent capabilities onboard terminals [2]. With the rapid progress of current artificial intelligence technologies, a large number of intelligent applications have emerged, including collaborative navigation systems, vehicle collision detection, and augmented reality/virtual reality [3,4]. These applications impose stringent requirements on task delay and energy consumption, posing significant challenges for resource-constrained in vehicle computing platforms [5]. To address this, Vehicle Edge Computing (VEC) has emerged as a pivotal solution. By constructing network topological relationships through vehicles, Roadside Units (RSUs) [6], tasks can be offloaded to RSUs for computation in a vehicle-to-infrastructure (V2I) manner, thus meeting the stringent requirements of intelligent applications [7]. This allows vehicles to obtain efficient and secure services [8].

Facing a large number of heterogeneous tasks from vehicles, RSUs struggle with inadequate computing resources. In response, a joint cloud computing (CC) approach can be used to offload computation tasks from RSUs to cloud servers [9]. Utilizing the powerful computing power of the cloud, RSU computational pressure is alleviated, and the Quality of Experience (QoE) of users can be improved [10].

The specific programs required for edge device computing tasks are usually defined as service caches. Taking vehicle environment perception as an example, the input data includes sensor data such as radar, cameras, and lidar around the vehicle. The execution of the task requires caching the corresponding environment perception service program into the vehicle or edge server, whose input data is usually unique and difficult to reuse [11], but the relevant service program data can be reused in future executions of similar tasks.

For RSUs, they need to parallel process significant volumes of vehicular requests, and the vehicle density on the road is random [12], which results in dynamically changing computational resources [13]. Given constrained RSU storage capacity, only a portion of service programs can be hosted locally at a time, making selective caching imperative. In addition, computing resources across RSUs often exhibit disparate allocation dynamically, thus requiring collaborative offloading to balance the computing load. Since service cache decisions are closely related to computing offloading decisions, when a service program has already been cached on RSUs, we tend to choose to offload the execution of tasks. However, existing approaches fail to resolve critical challenges: 1) How to optimize conflicting objectives such as delay, energy consumption, load balancing, and privacy entropy in a dynamic vehicular environment. 2) How to jointly optimize service caching and computation offloading decisions. 3) Adaptive adjustment of the weights among objectives in multi-objective optimization.

We propose a cloud-edge collaborative Internet of Vehicles (IoV) scenario aimed at optimizing computation offloading and service caching strategies. Considering the dynamic nature of network structures and real-time changes in task requests within road environments, we introduce a Multi-Objective Reinforcement Learning (MORL) algorithm based on Reinforcement Learning (RL). This algorithm leverages storage space and computing resources from vehicles and RSUs to collaborate with the cloud for computation offloading while ensuring the privacy and security of tasks. By integrating Radial Basis Function Networks (RBFN) with Chebyshev scalarization, our approach dynamically adjusts the weights among multiple objectives in multi-objective optimization. The contributions of this paper are summarized as follows:

1. We design a joint service caching and computational offloading model for cloud-edge collaborative IoV, formulating task delay, device energy consumption, RSU load balancing, and task privacy entropy as a multi-objective optimization problem, which is optimized in a unified manner.
2. We propose a Multi-Objective DDQN-Based Edge Caching and Offloading Algorithm (MODDQN-ECO), which integrates a Radial Basis Function Network (RBFN) with Chebyshev scalarization. This hybrid approach dynamically adjusts weights by learning the value changes among multiple objectives and can also incorporate predefined preference values to guide decision making.
3. Through extensive simulation experiments comparing with benchmark algorithms and recent multi-objective optimization methods, the proposed approach demonstrates superior performance across multiple metrics, including task delay, energy consumption, RSU computational load, and task privacy entropy.

The remaining parts of this paper are as follows. Section 2 surveys state-of-the-art VEC computational frameworks. Section 3 describes the system model in detail. Section 4 elaborates the Multi-Objective Reinforcement Learning (MORL) methodology. Section 5 evaluates the performance of the algorithm through simulation experiments. Section 6 concludes the research findings of this paper.

2 Related Work

This section summarize prior work on computation offloading in IoV. Bai et al. studied the vehicle task planning problem with priority constraints and proposed a new strategy heuristic algorithm based on topological sorting to minimize vehicle travel distance and quickly obtain near-optimal solutions [14]. Zhu et al. designed an improved NSGA-III algorithm by constructing models of latency and energy consumption in the Internet of Vehicles, achieving multi-objective joint optimization of system delay and energy consumption [15]. Hussain et al. established an optimization framework examining infrastructure deployment in vehicular fog networks for latency and energy reduction [16]. Materwala et al. proposed an energy-aware offloading strategy and used an evolutionary algorithm to optimize the energy consumption of edge cloud devices [17]. Sun et al. propose a predictive computation offloading method for vehicles, offloading computation tasks to edge servers using vehicle-to-vehicle (V2V) and V2I communication, then performing multi-objective optimization of energy consumption and delay using a genetic algorithm [18]. Wu et al.'s cloud-edge collaboration work establishes a multi-objective formulation minimizing processing delay, device energy consumption, and costs [19]. Cui et al. developed an edge and terminals collaborative offloading scheme with multi-objective optimization model, addressing task latency and energy efficiency [20].

Most existing research on IoV computation offloading has focused on using genetic algorithms. However, traditional MOEAs may not perform well in real-time changing road environments as they require significant computational time to find Pareto-optimal solutions. DRL, on the other hand, can make real-time decisions in dynamic vehicle environments, making it more suitable for optimizing problems in IoV. Moghaddasi et al. proposed a task offloading method based on Rainbow Deep Q-Network (DQN) to optimize task allocation in a three-tier architecture consisting of device-to-device (D2D), edge, and cloud computing [21] Lin et al. employed a DRL framework to optimize real-time offloading decisions between mobile vehicles and RSUs, significantly reducing vehicle delay [22]. Lu et al. investigated a multi-objective edge server deployment strategy based on DRL, which reduces task delay while improving IoV coverage and load-balancing capabilities [23]. Liu et al. defined peak and off-peak unloading modes, combining a simulated annealing genetic algorithm with DQN to alleviate vehicle system delay [24]. Geng et al. designed a distributed DRL algorithm in order to reduce the task delay and device energy consumption in the computational offloading problem [25]. To address the high computational complexity and poor real-time performance of genetic algorithms, we employ a DRL algorithm as the multi-objective optimization approach in the IoV model.

The aforementioned studies predominantly focus on optimizing individual metrics, with insufficient exploration of multi-objective collaborative optimization. Amir Masoud Rahmani et al. reviewed integration approaches of cloud computing, fog computing, and edge computing in the Internet of Things (IoT) environment, with a focus on the application of meta-heuristic algorithms in task offloading optimization. They pointed out that these algorithms effectively reduce system delay, energy consumption, and cost by optimizing task allocation, and also outlined current research challenges and future directions [26]. Yao et al. formulated the delay and energy consumption optimization problem in dynamic IoV computation offloading as a Markov Decision Process (MDP) to achieve dual minimization [27]. Liu et al. integrated convex optimization algorithms with Deep Reinforcement Learning (DRL) to effectively reduce both delay and energy consumption in computation offloading [28]. Qiu et al. designed a vehicular network model considering stochastic task arrivals, time-varying channels, and vehicle mobility, and proposed a deep reinforcement learning-based computation offloading and power allocation scheme to minimize the total delay and energy consumption in VEC networks [29]. Shang et al. studied the problem of offloading non-orthogonal multi-access computation with joint resource allocation and optimized the weighted posterior of energy consumption and delay [30]. Ullah et al. comprehensively reviewed the current development of multi-agent task allocation strategies and multi-agent reinforcement learning (MARL) methods in the Internet of Vehicles (IoV), highlighting the key roles of intelligent learning architectures, security issues, and computing platforms. They also identified the main challenges in current multi-agent task allocation, including scalability, complexity, communication overhead, resource allocation, security, and privacy protection, and proposed potential directions for future research [31]. Most research on IoV computation offloading simply weight task delay and energy consumption into scalar objectives, which may not satisfy the preference requirements of different offloading tasks. To address this, we design a constrained multi-objective reinforcement learning algorithm with dynamically adjusted weights, targeting multiple optimization goals including delay, load balancing, energy consumption, and task privacy entropy.

Caching the relevant programs required to execute computational tasks is referred to as service caching. Jointly optimizing service caching and computation offloading decisions can significantly improve the efficiency of task offloading, effectively reducing server computational pressure and the delay of intensive tasks. Yan et al. proposed a low-complexity heuristic algorithm with a service cache to control the computational offloading behavior [32]. Ko et al. propose a joint offloading and service caching strategy that considers heterogeneous service preferences, maximizing the total utility of the MEC system [33]. Zhu et al. designed an edge caching scheme for the Internet of Vehicles. By pre-dividing and caching content segments, it reduces the load on the central server and alleviates network pressure [34]. Tang et al. proposed a computational offloading scheme in conjunction with service caching [35]. In the above studies, some researchers explore the relationship between computation offloading and caching, but there are still some shortcomings. For example, Yan's MEC environment is significantly different from the VEC environment. Zhu et al. study content caching rather than service caching. Tang's research does not consider the problem of limited resources. This paper integrates edge intelligence to achieve the coordination of caching and computing resources under resource-constrained conditions.

In summary, regarding the multi-objective optimization research in IoV models, many scholars tend to employ multi-objective evolutionary algorithms to optimize various performance metrics. However, such algorithms often fall short in performance when dealing with dynamic changes in road environments. On the other hand, while reinforcement learning-based methods have been applied for model optimization, these studies mostly concentrate on enhancing individual performance indicators, with insufficient exploration into comprehensively addressing multi-objective optimization problems. Furthermore, despite the significant role of service caching mechanisms in improving system efficiency, recent research that adequately considers this aspect in model design remains relatively scarce. Therefore, we propose a dynamic adaptive multi-objective reinforcement learning algorithm for joint computational offloading and service caching in a cloud-edge collaborative IoV model. The algorithm uses an RBFN network to dynamically update target weights while optimizing for delay, load balancing, energy consumption, and task privacy entropy. This effectively solves the complex scheduling problem in cloud-edge collaboration, maximizing utilization of resource-constrained vehicles and RSUs infrastructure through cloud-assisted delay optimization. Table 1 summarizes the research contents of related work.

Table 1: Related work comparison

Research focus	Representative works	Key limitations
Based on evolutionary algorithms	[14–20]	High computational complexity and limitations in dynamic scenario applications
Based on reinforcement learning	[21–25]	Slight insufficiency in multi-objective cooperative optimization
Multi-objective cooperative optimization	[19,20,26–31]	Restricted by static weight mechanisms, making it difficult to meet the diverse requirements of offloading tasks
Cooperative research on service caching and offloading	[32–35]	Failure to jointly optimize service caching and computation offloading strategies, with insufficient consideration of resource constraints

3 System Model

3.1 Cloud-Edge Collaborative IoV Model

We study an IoV system consisting of vehicles, RSUs and the cloud. There are n vehicles traveling on the road, the edge devices are m RSUs, and the cloud consists of a cloud server, Fig. 1 is the main architecture.

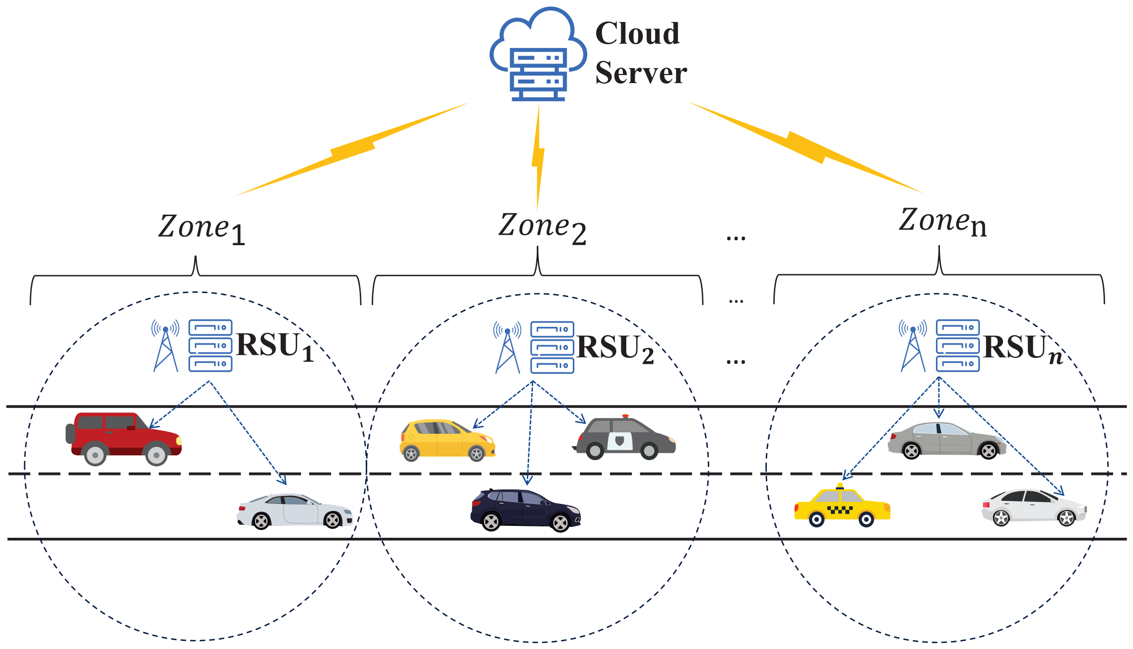

Figure 1: Cloud-edge collaborative IoV model

There are m RSUs located along the road, each covering an equal distance region. $V = \{V_1, V_2, \ldots, V_n\}$ denotes the set of vehicles, where computational tasks can be either processed locally or offloaded to RSUs for computation.

The $R = \{R_1, R_2, \ldots, R_m\}$ represents the set of distributed RSUs. Divide the time interval into $\{t_0, \ldots, t_i\}$ each of duration Δ t.update task offloading and service caching strategies in each interval. RSUs receive offloaded tasks from vehicles within their coverage. Then, RSUs can offload tasks to the cloud server based on the dynamic network environment. Cloud servers have sufficient computational resources to handle the offloaded computational tasks. Table 2 summarizes the main notations.

Table 2: Notation and definitions

Notations	Definitions
V	The set of vehicles
J	The set of tasks
R	The set of RSUs
C	Cache capacity
W	Transmission channel bandwidth
ω_0	Background noise
h	Channel gain
$P_{n,m}$	Vehicle to RSU transmission power
δ_n	Input data size for task n
τ_n	Storage space required for task n
f_v	Vehicle local CPU computing frequency
f_n	Frequency of computation assigned to task n
β	Energy consumed per CPU cycle
κ	Calculation of energy efficiency parameters
Off_R	Offloading ratio per task
T	The total delay of the task
E	Total energy consumption
K	Task Privacy Entropy
L	Load Balancing for RSUs

3.1.1 Task and Communication Model

We define the task corresponding to the vehicle as $J = \{J_1, J_2, \ldots, J_n\}$, where J_n contains the information $< \delta_n, \tau_n >$, which denotes the amount of data input to the computational task and the required storage space, respectively. Since parts of the task can be offloaded to RSUs or the cloud for execution, we define the offloaded variable as $Off_R(n) \in [0,1]$. Vehicles can perform some tasks locally and offload others for execution. Due to the mobile nature of vehicles in the road, setting the vehicle speed Sp_{min} to Sp_{max} [36], the collaborative edge offloading among multiple RSUs can further improve the efficiency of task execution.

The RSUs in each region can cover all vehicles within their region, where the total bandwidth of the RSUs in each region is W, with identical spectrum allocation and orthogonal channels assumed across distinct RSUs. Let $P_{n,m}$ be the transmission power of the vehicle, $h_{n,m}$ is the channel gain, which is an independent and identically distributed (i.i.d) random variable [37]. Num_J^m be the number of tasks received by R_m within a given time. The vehicle-RSU signal-to-noise ratio is derived as:

$$sinr_{n,m} = \frac{P_{n,m} h_{n,m}}{\frac{W}{Num_J^m}\omega_0} \tag{1}$$

where, ω_0 is the channel background noise. The transfer rate at which task J_n offloads to R_m is:

$$\eta_{n,m} = \frac{W}{Num_J^m} \log_2 \left(1 + sinr_{n,m}\right) \tag{2}$$

3.1.2 Caching Model

At time slot t, the vehicles within the range of the RSU update the task request, when the local or offloaded computation is also completed. Meanwhile, the system completes caching decisions on RSUs. The following is the constraint on the cache space:

$$\sum_{i=1}^{n} X_{J_i}^{V}(t)\delta_i \le C_V \tag{3}$$

$$\sum_{i=1}^{n} X_{J_i}^{R}(t)\delta_i \le C_R \tag{4}$$

where the binary variables $X_{J_i}^{V}(t) \in \{0,1\}$ and $X_{J_i}^{R}(t) \in \{0,1\}$ are used to denote whether or not the computational task J_n is cached on the vehicle V or RSU, and $X_{J_i}^{V}(t) = 1$ and $X_{J_i}^{R}(t) = 1$ denote that it has been cached. 0 means not cached. C_V and C_R denote the cache space of the vehicle and RSU, respectively.

As shown in Fig. 2, in this paper, the computing tasks of vehicles adopt a partial offloading strategy. When the relevant programs of the computing tasks requested by vehicle V_n have been cached at the RSU, the vehicle offloads the tasks to RSUs for execution through wireless transmission, and the offloading process is related to the location of the vehicle. If there is no cache of this computing task on RSUs, the task will be offloaded to the cloud for computation. Computing tasks choose different offloading strategies according to different task caching strategies. Therefore, it is necessary to reasonably allocate cache space and computing resources to maximize system revenue.

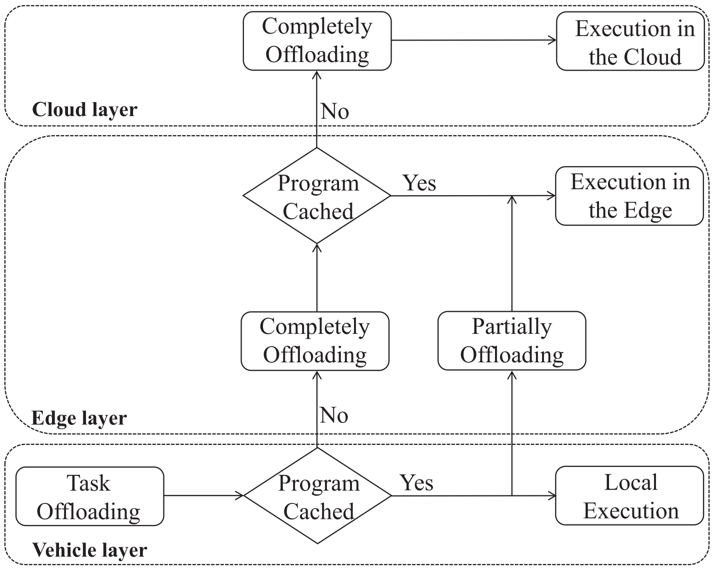

Figure 2: The computation offloading flowchart

3.2 Computational Model

3.2.1 Local Computational Model

The vehicle can leave the $(1 - Off_R(n))\delta_n$ portion of the task J_n to be executed locally and the $Off_R(n)\delta_n$ portion to be uploaded to RSUs for computation. Denoting f_V as the computational frequency of local vehicle execution and θ represents the computational intensity per data bit in CPU cycles. So the time to perform the task J_n locally [38]:

$$T_{loc}(n) = \frac{(1 - Off_R(n))\delta_n\theta}{f_v} \tag{5}$$

The computational consumption of locally executing tasks can be derived as follows, where β denotes the single-cycle energy consumption per CPU:

$$E_{loc}(n) = \beta(1 - Off_R(n))\delta_n\theta \tag{6}$$

3.2.2 Offloading Execution Mode

When the vehicle V_n offloads the computational task to nearby RSUs it goes through the following stages:

a) *Decision stage*: When vehicle V_n offloads a task to the nearest RSU, the system selects the offloading ratio of the task based on the service cache and also determines the allocation of computational resources.
b) *Transmission stage*: When the vehicle offloads the task to the corresponding RSU, then the time required to upload the task J_n is:

$$T_{off}^{up}(n) = \frac{Off_R(n)\ X_{J_n}^R\ \delta_n}{\eta_{n,m}} \tag{7}$$

c) *Processing stage:* The RSU processes the offloaded tasks based on the allocated computational resources, where f_n is the computational resource allocated by the RSU to task J_n. The time for the task offload calculation is:

$$T_{off}^{exe}(n) = \frac{Off_R(n)\ X_{J_n}^R\ \delta_n\ \theta}{f_n} \tag{8}$$

d) *Results return stage:* When the RSU completes the computation of task J_n, it returns the results to the corresponding vehicle. Since the computational results are significantly smaller than the input data size of the task, the result return time can be neglected [39].

With the above four stages, the total delay T_{off} for task offloading to RSU execution is:

$$T_{off}(n) = T_{off}^{up}(n) + T_{off}^{exe}(n) \tag{9}$$

The energy consumption for executing the offloaded task is calculated as follows [40]:

$$E_{off}^{exe}(n) = \kappa(f_n)^{\alpha}\ T_{off}^{exe}(n) \tag{10}$$

where κ and α are the computational energy efficiency parameter and power performance parameter, respectively. p_v is the power of the vehicle to send the task, then the total execution energy of the task J_n is [41]:

$$E_{off}(n) = p_v\ T_{off}^{up}(n) + E_{off}^{exe}(n) \tag{11}$$

When the vehicle and RSU have no cached tasks or insufficient computing resources, the RSU can offload tasks to the cloud. Due to the abundant computational resources available in the cloud, the delay of cloud computing can be negligible. Therefore, the computational delay of task J_n in the cloud can be simplified as:

$$T_{cloud}(n) = (1 - X_{J_n}^{V})(1 - X_{J_n}^{R})\left(\frac{\delta_n}{\eta_{n,m}} + \frac{\delta_n}{\eta_{cloud}}\right) \tag{12}$$

where η_{cloud} represents the data transfer rate from RSUs to the cloud. The energy computation for RSUs uploading data to the cloud server for further computation is formulated as:

$$E_{R}^{cloud}(n) = (1 - X_{J_n}^{V})(1 - X_{J_n}^{R})p_r\frac{\delta_n}{\eta_{cloud}} \tag{13}$$

where p_r is the power of the RSUs to send the task to the cloud.

3.2.3 Task Privacy

There is a possibility of data leakage when offloading some privacy-sensitive tasks on the vehicular terminal. Therefore, we use privacy entropy as an indicator of security in the task data transfer process [42]. When the value of privacy entropy is higher, it indicates that the transmission process of the task is more secure, and we set the task arrivals to conform to the Poisson distribution:

$$\xi_i = \frac{\lambda^{\varepsilon_i}}{\varepsilon_i!}e^{-\lambda}\ ,\ i = 1, 2, \dots, n \tag{14}$$

where $0 \le \xi_i \le 1$ and $\sum_{i=1}^{n} \xi_i = 1$, we denote the equivalence of the tasks J_n and ξ_i as:

$$\begin{pmatrix} J_n \\ \xi_n \end{pmatrix} = \begin{pmatrix} J_1, J_2, \dots, J_n \\ \xi_1, \xi_2, \dots, \xi_n \end{pmatrix} \tag{15}$$

The privacy entropy of J_n is:

$$K(J_n) = -\sum_{i=1}^{n} \xi_i \log_2 \xi_i \tag{16}$$

3.2.4 Computing Load

In this paper, since the Cache and computing resources of RSUs are unchanged, the rise of workload in some time periods will lead to a large fluctuation in the load balancing rate of RSUs, and the quality of service (QoS) of users deteriorates consequently, and optimization of the load balancing can effectively improve the resource utilization rate of RSUs [43]. We define the computational load of RSUs serving vehicle V_n at time t as:

$$L(t_n) = \frac{\sum_{n \in N_m} Off_R(n)\ N_m\ X_{J_n}^{R}\ \delta_n\ \theta}{f_n T} \tag{17}$$

where N_m denotes the number of vehicles in the service range of R_m. In this system, the computational resources of RSUs should be reasonably allocated to avoid the shortage of computational resources of some RSUs, while maintaining the system operation stability of RSUs. Therefore, a lower L value can make the computational resources of RSUs in the system relatively balanced, reduce the congestion of some RSUs due to too many tasks, and keep the delay at a relatively low level.

3.3 Problem Formulation

Based on the above introduction of the local and offloaded computing models, the delay and energy consumption of J_n are:

$$T(n) = \max\{(T_{loc}(n) + T_{off}(n)), T_{cloud}(n)\} \tag{18}$$

$$E(n) = E_{loc}(n) + E_{off}(n) + E_R^{cloud}(n) \tag{19}$$

This study aims to minimize application delay, reduce energy consumption and the computational load on Roadside Units (RSUs), while balancing user experience and maximizing privacy entropy. That is, formulas (16)–(19). A conflicting relationship exists among these objectives: reducing delay inevitably requires consuming more computational resources, leading to increased energy consumption on devices, and maximizing privacy entropy introduces additional task transmission delay. To address this, this paper establishes a multi-objective framework through the joint optimization of computation offloading and service caching:

$$min\{T, E, L\, and\ -K\}, \tag{20}$$

s,t:

$$\begin{array}{ll}
0 \le Off_R(n) \le 1, & C1 \\
X_{J_n}^V \in \{0,1\}, & C2 \\
X_{J_n}^R \in \{0,1\}, & C3 \\
f_n \in (0, f_{max}], Off_R(n) \ne 0, & C4 \\
\sum_{i=1}^{Num_J^m} X_{J_i}^R \tau_i \le C_R, & C5 \\
\sum_{n=1}^{N} \tau_i \le D_\Delta, & C6 \\
n \in N, & C7
\end{array} \tag{21}$$

where constraint $C1$ specifies the range of offloading variables. Constraints $C2$ and $C3$ specify the caching decision variables. Constraint $C4$ specifies the computational resources allocated to tasks offloaded to RSUs. Constraint $C5$ specifies that the total number of tasks cached on each RSUs cannot exceed the storage ceiling capacity. Constraint C6 ensures the total storage space occupied by cached services does not exceed the maximum cache capacity D_Δ. Constraint $C7$ specifies a range for the number of vehicles.

3.4 Extreme Value Analysis for T and E

The maximum and minimum values of T and E are first evaluated for different cases.

1) *Case1*:

$$T_{case1}(n) = \frac{\delta_n \theta}{f_v} \tag{22}$$

2) *Case2*:

$$T_{case2}(n) = \max\left\{1 - Off_R(n)\frac{\delta_n \theta}{f_v}\ ,\ Off_R(n)\left(\frac{\delta_n}{\eta_{n,m}} + \frac{\delta_n \theta}{f_n}\right)\right\} \tag{23}$$

3) *Case3*:

$$T_{case3}(n) = \frac{\delta_n}{\eta_{n,m}} + \frac{\delta_n \theta}{f_n} \tag{24}$$

4) *Case4*:

$$T_{case4}(n) = \frac{\delta_n}{\eta_{n,m}} + \frac{\delta_n}{\eta_{cloud}} \tag{25}$$

It can be concluded from the above:

$$\max\{T(n)\} = \max\{T_{case1}(n), T_{case4}(n)\} \tag{26}$$

$$\min\{T(n)\} = \min\{T_{case2}(n), T_{case3}(n)\} \tag{27}$$

Since Case2 has the nature of a segmented function, we can derive its minimum delay:

$$\min\{T_{case2}(n)\} = \frac{\delta_n(f_n + \eta_{n,m})}{f_v f_n + \eta_{n,m}(f_v + f_n)} \tag{28}$$

where $Off_R(n) = 1/(1 + f_v(\frac{1}{\eta_{n,m}} + \frac{1}{f_n}))$. Thus the minimum value of $T(n)$ can be expressed as:

$$\min\{T(n)\} = \min\{T_{case2}(n)\} \tag{29}$$

From the above it can be obtained that the delay of the task is related to the communication, computation, and caching capabilities of the device. Therefore, in order to minimize the delay, one should try to avoid choosing the decision that will lead to the highest delay.

Similarly, the maximum and minimum values of $E(n)$ need to be evaluated under different caching scenarios.

1) *Case1*:

$$E_{case1}(n) = \beta \delta_n \theta \tag{30}$$

2) *Case2*:

$$E_{case2}(n) = \beta(1 - Off_R(n))\delta_n \theta + p_v \frac{Off_R(n) X_{J_n}^R}{\eta_{n,m}} + \kappa(f_n)^\alpha \frac{Off_R(n) X_{J_n}^R \beta}{f_n} \tag{31}$$

3) *Case3*:

$$E_{case3}(n) = p_v \frac{\delta_n}{\eta_{n,m}} + \kappa(f_n)^\alpha \frac{\delta_n \beta}{f_n} \tag{32}$$

4) *Case4*:

$$E_{case4}(n) = p_v \frac{\delta_n}{\eta_{n,m}} + p_r \frac{\delta_n}{\eta_{cloud}} \tag{33}$$

Based on the above, we can conclude:

$$\max\{E(n)\} = E_{case3}(n) \tag{34}$$

$$\min\{E(n)\} = E_{case4}(n) \tag{35}$$

By performing an extreme value analysis on T and E, it is clear that in order to minimize the processing delay of the on-board tasks, the decision scheme of Case2 and Case3 should be chosen as much as possible, i.e., the tasks should be processed as much as possible in collaboration with the RSUs.

4 Service Caching and Offloading Algorithms Based on Multi-Objective Reinforcement Learning

In this paper, Simultaneous optimization of multiple factors task delay, load balancing rate, and privacy entropy is required. Regarding multi-objective optimization (MOO), evolutionary algorithms remain the predominant approach in existing studies. However, their slow convergence speed impedes applicability in real-time dynamic vehicular offloading scenarios. DRL emerges as an effective methodology for dynamic environment optimization, maximizing expected cumulative rewards to achieve objectives. It aggregates demand-resource information within vehicular networks, subsequently executing actions to optimize joint offloading policies alongside resource distribution.

We formulate the problem as a Markov Decision Process (MDP) and propose an multi-objective Reinforcement Learning algorithm combining RBFN network with Chebyshev method. By dynamically adjusting the weights of objectives including task delay, device energy consumption, device computation load, and privacy entropy, find the optimal caching and offloading strategy.

4.1 Markov Decision Process

An MDP in the usual case is defined as a quintuple $< S, A, P, R, \gamma >$, where corresponding sequentially to the state space, action space, transition probability, reward function, and discount factor, respectively. The MDP formulation for our problem is defined as follows:

- *State Space*: We denote the state at a t time as $s(t) = \{\Phi(t), C(t), X^R(t), X^V(t), h(t)\}$, where $\Phi(t)$ denotes the currently available computational resources, $C(t)$ denotes the current remaining storage space, $X^R(t)$ and $X^V(t)$ represent the cache variables of the vehicle and RSU at time t, respectively, and $h(t)$ is the changing channel state.
- *Action Space*: The agent makes a decision strategy based on $s(t)$, which is to choose the action $a(t)$. $a(t) = \{X^R(t), V_{id}, Off(t), f(t)\}$, where $X^R(t)$ stands for the cache variable of RSUs after completing the task offloading and computation in t time, V_{id} is the identification of the vehicle, $Off(t)$ is the percentage of the vehicle's v offloading the task to the edge server in t time, and $f(t)$ denotes the computing resources allocated to the task. proportion, and $f(t)$ denotes the computational resources allocated to the task.
- *Reward*: The reward at time t is set to $r(t)$.
- *Transition probability:* The probability of the environment moving from $s(t)$ to $s(t+1)$ at the next time t is denoted as $P\{s(t+1)|s(t)|s(t), a(t)\}$.

We complete the optimization problem by learning the policy π to maximize the expected cumulative return Υ^{π}:

$$\Upsilon^{\pi} = \max_{\pi} \mathbb{E}_{\pi} \left\{ \sum_{t} \gamma(t) r[s(t), a(t), s(t+1)] \right\} \tag{36}$$

where $\gamma(t)$ is the discount factor, $\mathbb{E}_{\pi}$ is the desired strategy π, and $r[s(t), a(t), s(t+1)]$ is the reward. The strategy that maximizes the gain to the system is obtained by searching the strategy space, representing The reward value Q for the state-action is [44]:

$$Q^{*} = r(s, a) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^{*}(s', a') \tag{37}$$

4.2 Multi-Objective DRL

Compared to standard single-objective reinforcement learning, multi-objective reinforcement learning [45] can obtain multiple reward values as feedback from the environment when an agent takes an action. In this paper, the task delay, device energy consumption, privacy entropy and computing load rate of vehicular tasks are all reward values. Therefore, the feedback reward is a vector corresponding to multiple objectives, rather than a scalar value:

$$\vec{r} = \{r_1, r_2, r_3, r_4\} = \{T, E, L, -K\} \tag{38}$$

Since there are conflicting relationships among the multiple objectives to be optimized in this paper, the algorithm needs to find the strategy that maximizes the gain of the system among these conflicting relationships. We use a nonlinear scalarization method to solve this multi-objective optimization problem.

4.2.1 Multi-Objective DRL Based on Chebyshev's Scalarization Method

Considering the issue that linear weighting method cannot approximate all Pareto frontiers, we adopt the Chebyshev scalarization method, taking the ideal point z^* as the reference point and using L_p norm as the distance metric. We assign a coefficient w_i to each objective function and measure the value of each objective function o_i to obtain the performance of each objective function in the multi-objective optimization process.

$$\min L_p(x) = \left(\sum_{i}^{m} w_i |o_i(x) - z_i^*|^p \right)^{\frac{1}{p}} \tag{39}$$

The *Chebyshev* scalarization is represented as:

$$\min L_{\infty}(x) = \max_{i=1,\ldots,m} w_i |o_i(x) - z_i^*| \tag{40}$$

where $1 \le p < \infty$, $w_i \ge 0$, and $\lim_{p \to \infty}$. The value of Q for *Chebyshev* scaling is then obtained as:

$$Q^{cs}(s, a) = \max_{i=1,\ldots,m} w_i |Q(s, a) - z_i^*| \tag{41}$$

Intelligent bodies make the value of the action constantly close to the optimum, and eventually the system is able to obtain the optimal computational offloading gain.

4.2.2 Multi-Objective DDQN Computational Offloading Algorithm

In multi-objective double deep Q-network (DDQN) reinforcement learning computational offloading algorithm, where RBFN network is used to learn multiple different objective values to better adapt to weight changes, while also better addressing the Q-value overestimation issue in DQN. As shown in Fig. 3.

Each DDQN is denoted by DDQNT, DDQNE, DDQNL, and DDQNK, and its intelligences are rewarded based on different optimization objectives. In this paper, we set the reward function based on delay, energy consumption, computational load, and privacy entropy factor sub-objectives separately to obtain a better multi-objective solution. We correspond to the value of each objective by defining a dynamic weight $w_i \in [0, 1]$, so the Q value of the dynamic weight is:

$$Q_w = \sum_{i=1}^{M} w_i \, scale(\vec{Q^*}) \tag{42}$$

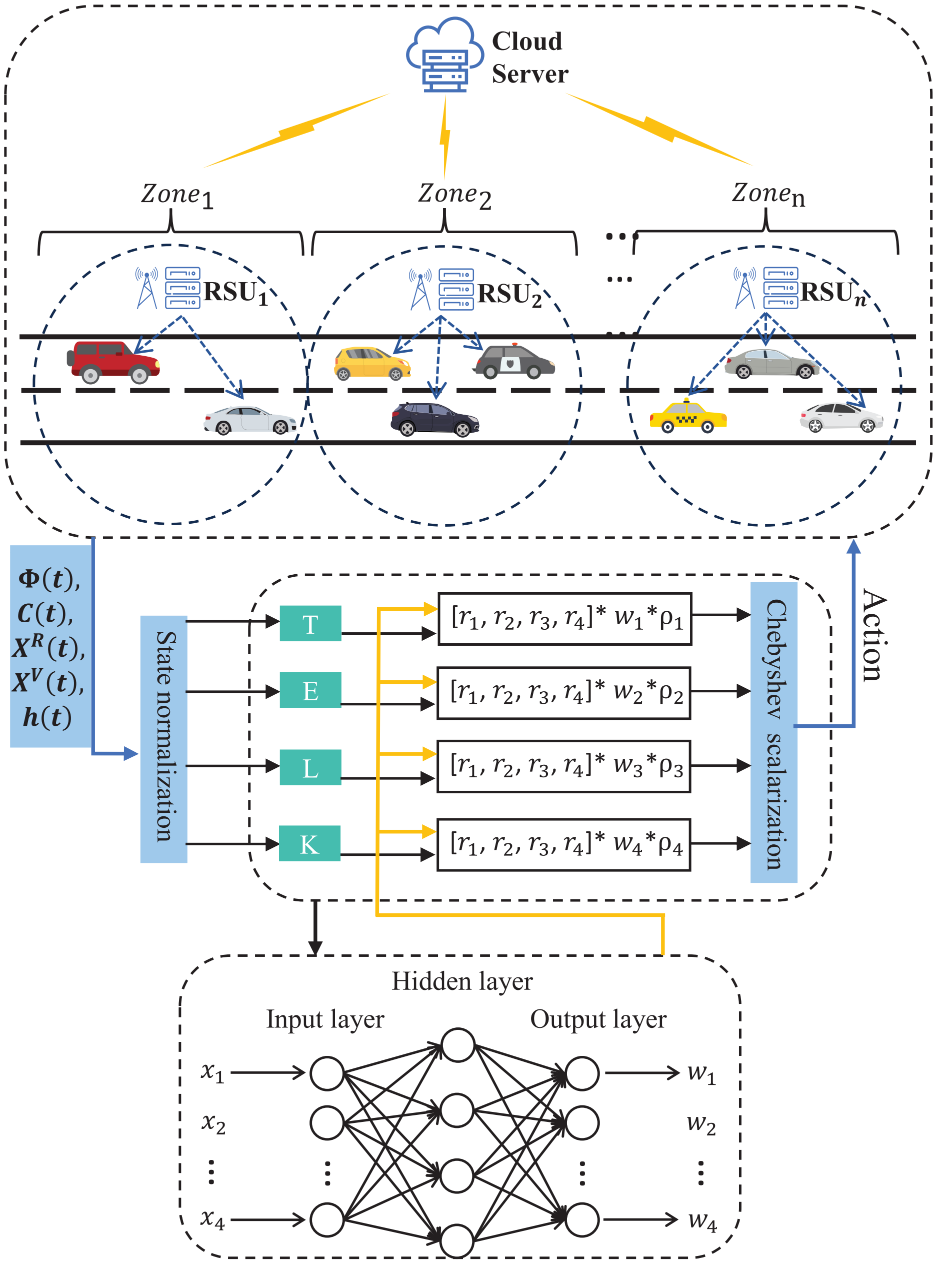
Cloud Server
$Zone_1$
$Zone_2$
$Zone_n$
RSU_1
RSU_2
RSU_n
$\Phi(t)$, $C(t)$, $X^R(t)$, $X^V(t)$, $h(t)$
State normalization
T
E
L
K
$[r_1, r_2, r_3, r_4]^* w_1{}^*\rho_1$
$[r_1, r_2, r_3, r_4]^* w_2{}^*\rho_2$
$[r_1, r_2, r_3, r_4]^* w_3{}^*\rho_3$
$[r_1, r_2, r_3, r_4]^* w_4{}^*\rho_4$
Chebyshev scalarization
Action
Hidden layer
Input layer
Output layer
x_1
x_2
x_4
w_1
w_2
w_4

Figure 3: MODDQN-ECO algorithmic frameworkm

We denote σ_i as additional preference weights that can be set according to different preferences, e.g., the preference value of privacy entropy can be appropriately increased when performing certain types of data-sensitive tasks. The Q_w^* of preference setting combined with dynamic weights is denoted as:

$$Q_w^* = \sum_{i=1}^{M} \sigma_i w_i \, scale(\vec{Q^*}) \tag{43}$$

With Q_w^*, the agent will select strategies with higher reward values in order to satisfy the multi-objective optimization.

4.2.3 RBF Neural Network Update

The RBFN network [46], consisting of an input layer, a hidden layer, and an output layer as shown in Fig. 3, is a feedforward neural network. It has a large number of hidden neurons and can accurately approximate continuous functions. The Gaussian function of the RBFN network is defined:

$$G(\|x - c\|, \chi) = exp\left(-\frac{\|x - c\|^2}{2\chi^2}\right) \tag{44}$$

where $x = [x_1, x_2, \ldots, x_d]$ denotes the d-dimensional input vector, and the value of the real-valued RBF function depends on the distance to any point c, As the difference between x and c increases, the function value decreases accordingly [47]. Here, ρ_i denotes the weight from the i-th hidden node to the output node, and the parameter vector of the RBF G_i is represented by S_i. The output of the network's hidden unit is then given by:

$$g(x) = \sum_{i=1}^{M_R} \rho_i G_i(x; S_i) \tag{45}$$

M_R denotes the number of hidden layer nodes. To find the optimal strategy, the RBFN influences the decision-making process through a dual-stage mechanism: first, in the weight generation layer, the network output $g(x)$ is normalized to generate weights w_i; subsequently, in the preference fusion layer, these weights w_i are combined with additional preference weight parameters σ_i to form the final preference weights Q_w^*. This dual-stage mechanism enables the system to adapt to user-defined requirements. During the decision optimization process, the MODDQN-ECO algorithm learns from sampled data updates and employs the Chebyshev scalarization method to adjust the Q-values of action a across different objective functions. Subsequently, the RBFN network dynamically updates the target weights.

4.2.4 Learning Process

Each DDQN corresponds to a Q function for the optimization objective, in which $Q_j^{target} = r_i + \gamma Q^-(argmax_{a'} Q(a', s_{i+1}), s_{i+1})$ is used instead of $Q_j^{target} = r_i + \gamma Q^-(max_{a'} Q(a', s_{i+1})$ to update the Q value, which is used to solve the overfitting of Q value in DQN. Two value functions in the algorithm, one for action selection and one for evaluating the value of the current state, are parameterized as w and w'. The loss function for each DDQN for iterative optimization is as follows:

$$loss(w) = \frac{1}{Num_s} \sum_{j=1}^{Num_s} (Q_j^{target} - Q(s_j; a_j; w_j))^2 \tag{46}$$

where Num_s is the number of randomly selected samples from the replay buffer. An overview of the MODDQN-ECO algorithm is described in Algorithm 1 by gradually minimizing the loss function value $loss(w)$ to bring it close to the target value.

Algorithm 1: Multi-objective DDQN-based edge caching and offloading algorithm

Input: Number of iterations ep_{max}, state space S, action space A, current network Q, target network Q', number of gradient descent samples Num_s, preference weights σ_i, reference point z^*.
Output: Optimal Network Parameters

```
1: **Initialization:** Reset replay buffer and weight history
2: **for** $episode \rightarrow 1\ to\ ep_{max}$ **do**
3:     **while** $t \leq T$ **do**
4:         agent randomly selects the initial action $a(t)$ from the action space with probability $\phi$.
5:         Otherwise, $a(t)$ is selected according to $max_a Q^*(s(t), a; w_t)$
6:         **if** $a(t)$ satisfies the constraints $C1 - C6$ **then**
7:             Chebyshev scalarization method for scaling Q-values;
8:             Observe the scaled reward $r(t)$ and preprocess the next state $s(t+1)$;
9:             Store $(s(t), a(t), r(t), s(t+1))$ in replay buffer
10:            Store transition $(s, a(t), r(t), s')$ into the replay buffer, $s(t) \leftarrow s(t+1)$
11:            Randomly sample $(s_j, a_j, r_j, s_{j+1})$ in the replay buffer and compute the target $Q$ values $Q_j^{target}$
12:            **if** $s' = s_{terminal}$ **then**
13:                $Q_j^{target} = r_j$
14:            **else**
15:                $Q_j^{target} = r_j + \gamma Q'(s(t+1) arg\max_{a'} Q(s_{j+1}, a; w_t'))$
16:            **end if**
17:            Perform gradient descent on $\frac{1}{Num_s} \sum_{j=1}^{Num_s} (Q_j^{target} - Q(s_j, a_j, w_t))^2$.
18:            **if** $t\%\zeta = 1$ **then**
19:                Update target network parameters $w' = w$
20:            **end if**
21:            **if** $s(t+1)$ is in terminated state **then**
22:                Current iteration is complete, otherwise proceed to step 4
23:            **end if**
24:        **end if**
25:    **end while**
26: **end for**
```

5 Performance Evaluation

5.1 Experimental Settings

In this section, we experimentally evaluate the performance of the proposed MODDQN-ECO algorithm in the IoV model. Previous studies [48,49] provide empirical references for parameter settings, with detailed experimental parameters summarized in Table 3. We implement the IoV simulation environment using TensorFlow 1.15.0 and Python 3.7. The simulation establishes a 1000-meter-long two-lane road with five roadside servers deployed, each covering a 200-meter range to divide the road into five segments. Task sizes are set within [20, 25] Mb and vehicle CPU frequencies are configured at $1.5 * 10^9$ Hz. The time slot duration in the simulation is carefully designed to ensure both task offloading completion and sufficient decision-making time for the agent regarding service caching and offloading strategies. We compare the MODDQN-ECO algorithm with the following algorithms:

Table 3: Simulation parameters

Parameter description	Value
The set of vehicles (V)	[10, 40]
Vehicle CPU cycle (f_V)	$1.5 * 10^9$ cycles/s
Edge server CPU cycles (f_{max})	$3 * 10^{10}$ cycles/s
Computation intensity (θ)	50 cycle/bit
Edge server bandwidth (W)	20 MHz
Storage space required each task (ι_n)	50 Mb
Edge server storage capacity	100 Mb
Number of task types	5
Energy consumption of each CPU cycle (β)	4 J/G cycles
The computing energy efficiency parameters (κ)	$1 * 10^{-28}$
Exponent parameter (α)	3
The transmitted power of Vehicle $P_{n,m}$	1 W
Transmission rate between edge servers (η_{RSU})	40 Mbps
Transmission rate from edge to the cloud (η_{cloud})	10 Mbps
Transmission power from edge servers to the cloud (p_r)	2 W
Channel background noise (ω_0)	−173 dBm
Discount factor (γ)	0.99
Frequency of target network updates (ζ)	200
Batch size (Num_s)	64
Replay buffer	5000

Multi-objective DQN Computational Offloading Algorithm (DQN) [50]: multi-objective DQN computational offloading algorithm using linear scalarization method is very reliable to solve computational offloading problems in dynamically changing environments using DQN. Its approximates the value function through a convolutional neural network. agent obtains the corresponding Q-value based on the state of the IoV environment and selects an action using a greedy strategy, then updates the network parameters of the value function based on the feedback reward value and the new state. The intelligents use an empirical playback method [51] to improve data efficiency and enhance training stability.

Improved Non-dominated Sorting Genetic Algorithm (INSGA-II) [52]: An improved NSGA-II algorithm is employed to solve the model in this paper. The algorithm retains the fast non-dominated sorting mechanism. In the genetic operations, it adopts Simulated Binary Crossover (SBX) and Simulated Binary Mutation (SBM) to generate offspring with good diversity. This enhancement effectively improves the convergence speed of the algorithm, the uniformity of solution distribution, and the robustness in handling complex multi-objective optimization problems, making it well-suited for the conflicting optimization objectives addressed in this work.

Cloud Computing (CO): all on-board tasks are offloaded to the cloud to perform computation.

Random offload policy (RO): service caching and computation offload policies are randomly generated for each time period.

Fig. 4 shows the reward curve of MODDQN-ECO algorithm, and the result verifies that MODDQN-ECO algorithm can converge quickly while satisfying multiple objective balances, and then get the optimal offloading gain value. And it requires fewer iterations than some classical evolutionary algorithms, which can better cope with the time-varying road environment.

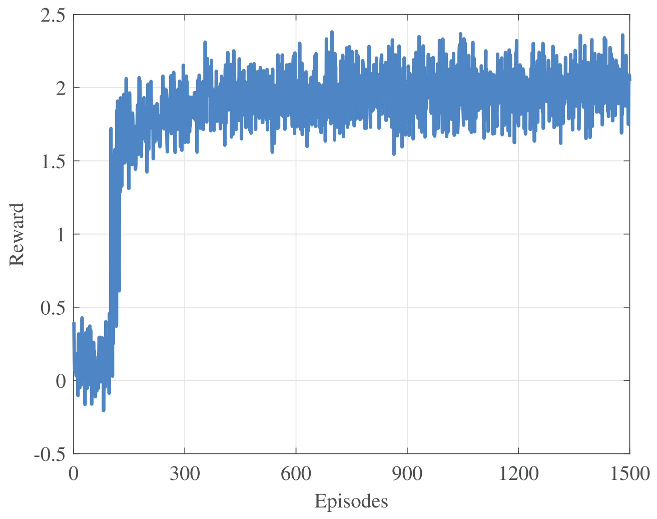

Figure 4: This is a figure example. Please remove all non-English terms or add a definition for them

5.1.1 Influence of Vehicle Number on IoV System

Fig. 5 experimentally evaluates algorithmic performance under varying vehicular density. In Fig. 5a,b, when the total number of vehicles on the road is relatively low, the performance gap between the MODDQN-ECO algorithm and DQN is not significant, and both tend to maintain balance across multiple objectives. However, they outperform the evolutionary algorithm INSGA-II in most scenarios, which is attributed to the faster convergence characteristics of reinforcement learning algorithms under the same number of iterations. MODDQN-ECO dynamically adjusts the weighting of each objective through the RBF network, enabling it to achieve a more balanced advantage in delay performance. This mechanism allows the algorithm to rapidly adapt to the environment and derive superior offloading decisions. As the number of vehicles increases, the task volume rises accordingly, leading to reduced per-vehicle bandwidth allocation and decreased task transmission rate, which consequently increases the task offloading delay. When the number of vehicles further increases, the delay advantage of task execution by RSUs weakens. At this point, the algorithm employs two key strategies to cope with the pressure: (1) Offloading some tasks to the cloud server to alleviate the computational resource pressure on the RSUs; (2) Some tasks are forced to execute locally. Although these strategies result in an increase in total task execution delay and device energy consumption, the rate of increase for MODDQN-ECO is significantly lower than that of the comparative algorithms. In high-density scenarios, by sensing the intensity of resource competition in the environment, the algorithm automatically adjusts the weight distribution among the various objectives, maintaining its performance advantage in terms of both delay and energy consumption. Compared to INSGA-II and DQN in terms of delay performance, MODDQN-ECO achieves average improvements of 10.64% and 8.71%, respectively. In terms of energy consumption, it shows improvements of 5.05% and 3.87% over the two algorithms, respectively.

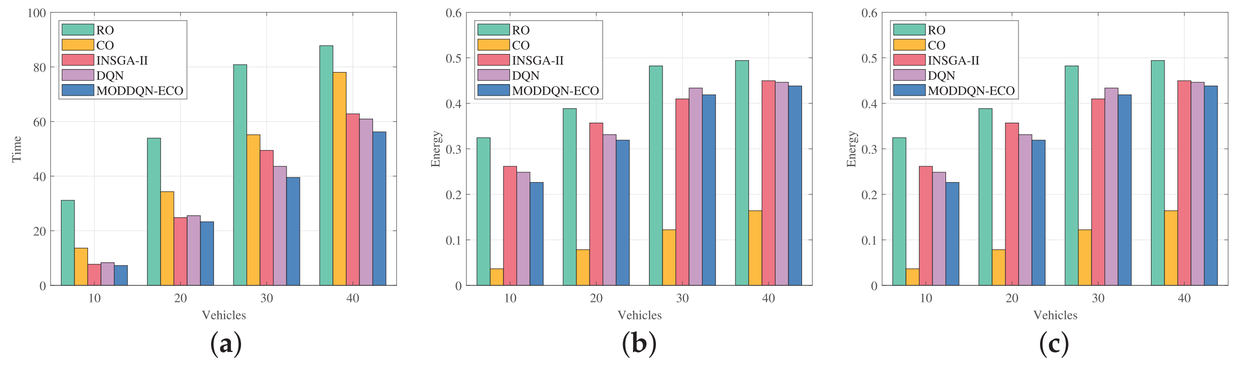

Figure 5: Performance of MODDQN-ECO algorithm with different number of vehicles

Fig. 5c demonstrates the performance of RSU load-balancing. MODDQN-ECO, under different vehicle densities, can dynamically adjust the weighting for load balancing to steer tasks toward underutilized nodes, exhibiting superior load-balancing capability. Compared to INSGA-II and DQN, it achieves average improvements of 11.07% and 3.11%, respectively. The MODDQN-ECO algorithm proposed in this paper significantly outperforms NSGA3 and DQN on key performance metrics such as delay and energy consumption. Furthermore, this advantage expands with an increasing number of vehicles, validating the effectiveness of its dynamic weighting mechanism in high-density IoV scenarios. Table 4 shows the core data of Fig. 5. The MODDQN-ECO algorithm proposed in this paper significantly outperforms the INSGA-II and DQN algorithms in key performance metrics, and this advantage becomes larger with the number of vehicles.

Table 4: Comparison of the proposed algorithm MODDQN-ECO with INSGA-II and the DQN algorithm across varying numbers of vehicles as the variable

Algorithms	INSGA-II	DQN	MODDQN-ECO	Vehicles
Time	7.7235958	8.3143065	**7.2632851**	10
Energy consumption	0.2619119	0.2480470	**0.2262327**	
Load balancing	0.1690597	**0.1521047**	0.1583245	
Time	24.810214	25.528019	**23.286735**	20
Energy consumption	0.3563106	0.3312175	**0.3190291**	
Load balancing	0.1301562	0.1080195	**0.0942891**	
Time	49.427315	43.594052	**39.537031**	30
Energy consumption	**0.4096829**	0.4331594	0.4187370	
Load balancing	0.0825419	**0.0720663**	0.0770381	
Time	62.818429	60.912037	**56.216951**	40
Energy consumption	0.4496572	0.4452631	**0.4381160**	
Load balancing	0.0407282	0.0432049	**0.0386172**	

5.1.2 Performance of Task Privacy Entropy

Fig. 6 verifies the MODDQN-ECO algorithm's performance in terms of privacy entropy. When the vehicle number reaches 40, offloaded tasks are split into more parts, increasing transmission time and delay. Despite this, the algorithm maintains low delay and high privacy entropy in the mid and late iterations. This shows that the MODDQN-ECO with the Chebyshev scalar approach can dynamically approximate optimal decisions in multi-objective optimization. In contrast, DQN and INSGA-II with fixed weights fail to effectively approach the Pareto front, resulting in lower privacy entropy.

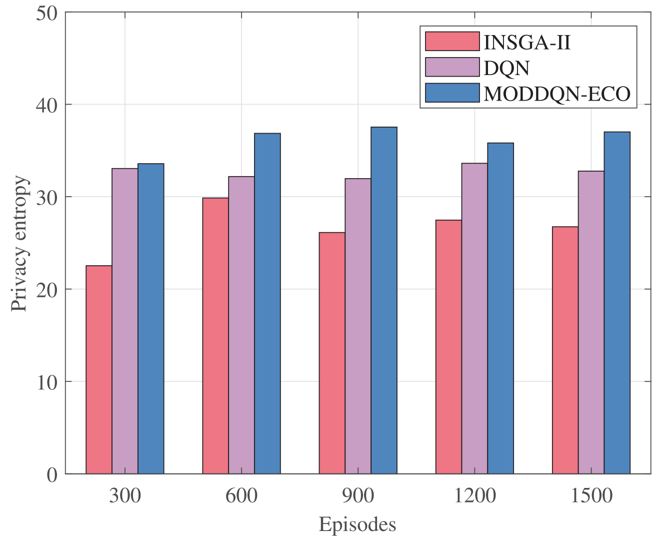

Figure 6: Task privacy entropy performance for different number of iterations

5.1.3 Performance under Different Task Sizes

In Figs. 7 and 8, we conduct an in-depth investigation into the impact of task size on delay and energy consumption. Fig. 7 illustrates the trend of task delay vs. task size, revealing a linear increase attributable to the proportional relationship between delay and input data volume. As task scales expand, the surge in data transmission and computational demands elevates overall system delay. To mitigate this, the system dynamically increases cloud offloading ratios, alleviating computational pressures on RSUs and vehicles while reallocating freed resources to accelerate processing of tasks already offloaded to RSUs—thereby achieving synergistic optimization and reducing delay impacts.

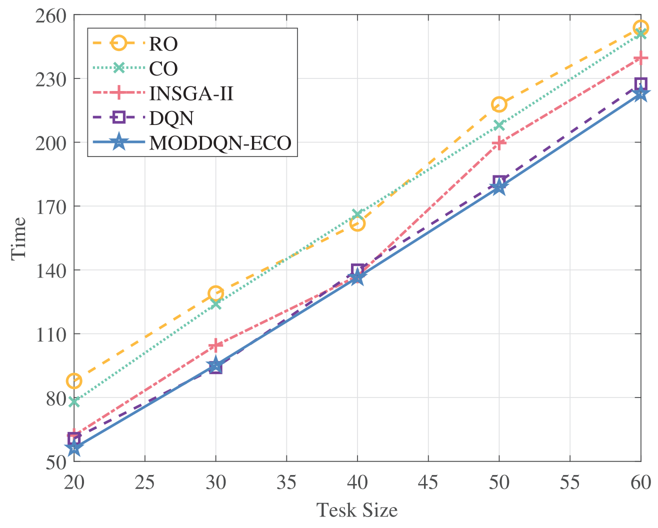

Figure 7: The performance of delay under different task sizes

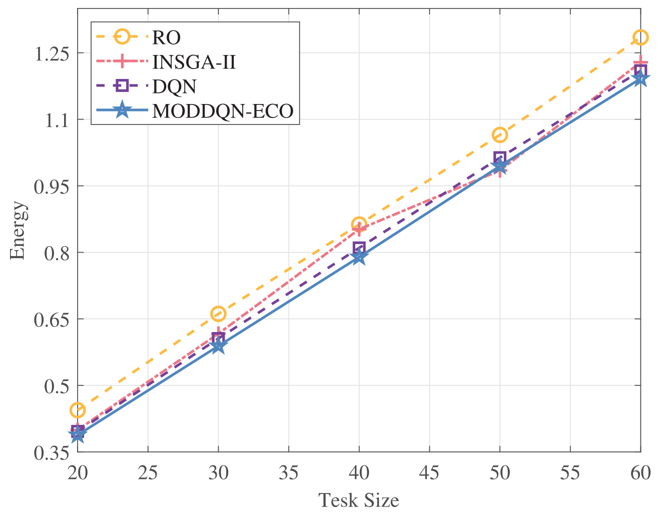

Figure 8: The performance of energy consumption under different task sizes

Fig. 8 further demonstrates a strong correlation between energy consumption and task scale. Larger tasks substantially increase data transmission and computational energy costs. Under these conditions, both DQN and MODDQN-ECO algorithms effectively orchestrate three-tier computing resources (local vehicles, RSUs, and cloud) to collaboratively fulfill offloading requests, yielding significant energy consumption advantages. Moreover, MODDQN-ECO's core capability of dynamically adjusting multi-objective weights through its RBFN network enables sustained optimal balance between delay and energy consumption during task scaling variations. This approach significantly outperforms static-policy algorithms in maximizing offloading gains. The comprehensive improvement in delay and energy consumption achieves enhancements of 10.26% and 4.33% compared to INSGA-II and DQN, respectively.

5.1.4 Delay Performance at Different Cycle Frequencies

Fig. 9 illustrates the impact of RSU computational frequency on total task delay, with the number of vehicles fixed at 40. It is evident from the figure that higher RSU computational frequencies significantly reduce system delay. This is because RSUs can rapidly process a larger number of offloaded tasks, effectively shortening the task response cycle. Consequently, the algorithm preferentially offloads more tasks to edge servers for computation. In this optimized scenario, all three intelligent algorithms demonstrate improved delay performance. However, unlike other static strategy algorithms, MODDQN-ECO distinguishes itself through its dynamic objective weight adjustment capability, enabling greater adaptability to such conditions by re-prioritizing task offloading. This mechanism maximizes the utilization of RSU computing resources while optimizing task distribution ratios, thereby fulfilling the low-delay requirements of complex tasks.

5.1.5 Delay Performance at Different Speed Range

Fig. 10 illustrates the impact of vehicles traveling at various speed ranges on latency. In this section, the number of vehicles is set to 40, and the latency performance is compared for tasks where vehicle speeds fall within the ranges of [40–60] km/h, [60–80] km/h, and comprehensively [40–80] km/h. Observably from the graph, vehicles operating at lower speed ranges generally experience less delay. As vehicle speeds increase, some tasks may be offloaded or completed while the vehicle has already moved into another server's coverage area, thereby increasing inter-server communication time. All three intelligent algorithms exhibit minimal performance variation, attributed to the high efficiency of communication between servers. Compared to the classical evolutionary algorithm INSGA-II and the reinforcement learning algorithm DQN, the MODDQN-ECO algorithm demonstrates greater stability and more effectively minimizes latency associated with service caching and computational offloading in dynamic vehicular scenarios.

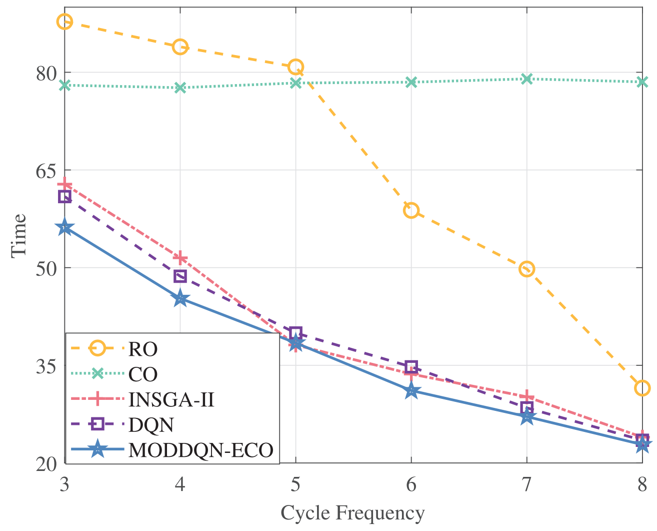

Figure 9: Total delay performance at different cycle frequencies of RSUs

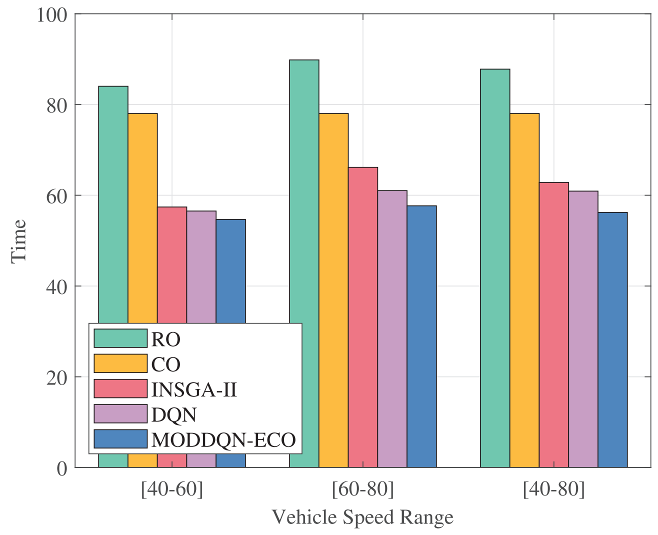

Figure 10: Delay performance at different speed range

6 Conclusion and Future Work

Aiming at the real-time varying vehicular network scenarios, this paper designs a novel computational offloading model and derives critical threshold values for task delay and energy consumption under different network conditions. By integrating delay, device energy consumption, RSU computing load, and task privacy entropy into a multi-objective optimization framework, the MODDQN-ECO algorithm with dynamic RBFN-based weighting achieves significant improvements: compared to INSGA-II and DQN, the average delay is reduced by 10.64% and 8.71%, while energy consumption decreases by 5.1% and 3.9%, respectively.

MODDQN-ECO effectively coordinates caching resources and RSU computing capabilities, maintaining robust performance during dynamic fluctuations in RSU frequency, thereby validating its efficiency.

Our approach still has certain limitations in terms of model architecture. For instance, it relies on the V2I communication paradigm for task transmission and does not fully utilize the task delivery capabilities of V2V communication, especially in forming service gaps within RSU coverage blind spots; the dynamic weighting mechanism may lead to policy instability in cases of extreme objective conflicts. In future research, we will explore homomorphic encryption-supported secure V2V task sharing protocols to enable safe reuse of cached tasks among vehicles; develop hierarchical scheduling mechanisms for concurrent computation-intensive tasks targeting vehicle clusters; and investigate federated learning approaches to enhance privacy protection capabilities in distributed edge environments.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by Key Science and Technology Program of Henan Province, China (Grant Nos. 242102210147, 242102210027); Fujian Province Young and Middle aged Teacher Education Research Project (Science and Technology Category) (No. JZ240101) (Corresponding author: Dong Yuan).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Junjun Ren; methodology, Guoqiang Chen and Zheng-Yi Chai; software, Guoqiang Chen; validation, Junjun Ren, Guoqiang Chen and Zheng-Yi Chai; formal analysis, Dong Yuan; investigation, Dong Yuan; resources, Junjun Ren; data curation, Dong Yuan; writing—original draft preparation, Guoqiang Chen and Zheng-Yi Chai; writing—review and editing, Junjun Ren; visualization, Zheng-Yi Chai; supervision, Junjun Ren; project administration, Junjun Ren; funding acquisition, Guoqiang Chen and Zheng-Yi Chai. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Luo Q, Li C, Luan TH, Shi W, Wu W. Self-learning based computation offloading for internet of vehicles: model and algorithm. IEEE Trans Wireless Commun. 2021;20(9):5913–25. doi:10.1109/twc.2021.3071248.
2. Zhai Y, Sun W, Wu J, Zhu L, Shen J, Du X, et al. An energy aware offloading scheme for interdependent applications in software-defined IoV with fog computing architecture. IEEE Trans Intell Transp Syst. 2021;22(6):3813–23. doi:10.1109/tits.2020.3044177.
3. Liu J, Guo H, Xiong J, Kato N, Zhang J, Zhang Y. Smart and resilient EV charging in SDN-enhanced vehicular edge computing networks. IEEE J Selected Areas Commun. 2020;38(1):217–28. doi:10.1109/jsac.2019.2951966.
4. Zhou H, Jiang K, Liu X, Li X, Leung VCM. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. IEEE Internet of Things J. 2022;9(2):1517–30. doi:10.1109/jiot.2021.3091142.
5. Malektaji S, Ebrahimzadeh A, Elbiaze H, Glitho RH, Kianpisheh S. Deep reinforcement learning-based content migration for edge content delivery networks with vehicular nodes. IEEE Trans Netw Serv Manage. 2021;18(3):3415–31. doi:10.1109/tnsm.2021.3086721.
6. Binh TH, Son DB, Vo H, Nguyen BM, Binh HTT. Reinforcement learning for optimizing delay-sensitive task offloading in vehicular edge-cloud computing. IEEE Internet of Things J. 2024;11(2):2058–69. doi:10.1109/jiot.2023.3292591.
7. Guo H, Zhou X, Liu J, Zhang Y. Vehicular intelligence in 6G: networking, communications, and computing. Veh Commun. 2022;33:100399. doi:10.1016/j.vehcom.2021.100399.
8. Hazarika B, Singh K, Biswas S, Li CP. DRL-based resource allocation for computation offloading in IoV networks. IEEE Trans Ind Inform. 2022;18(11):8027–38. doi:10.1109/tii.2022.3168292.

9. Chen C, Liu B, Wan S, Qiao P, Pei Q. An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. IEEE Trans Intell Transp Syst. 2021;22(3):1840–52. doi:10.1109/tits.2020.3025687.
10. Ju Y, Chen Y, Cao Z, Liu L, Pei Q, Xiao M, et al. Joint secure offloading and resource allocation for vehicular edge computing network: a multi-agent deep reinforcement learning approach. IEEE Trans Intell Transp Syst. 2023;24(5):5555–69. doi:10.1109/tits.2023.3242997.
11. Xu J, Chen L, Zhou P. Joint service caching and task offloading for mobile edge computing in dense networks. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications; 2018 Apr 15–19; Honolulu, HI, USA. p. 207–15.
12. Lyu F, Yang P, Wu H, Zhou C, Ren J, Zhang Y, et al. Service-oriented dynamic resource slicing and optimization for space-air-ground integrated vehicular networks. IEEE Trans Intell Transp Syst. 2022;23(7):7469–83. doi:10.1109/tits.2021.3070542.
13. Huang J, Wan J, Lv B, Ye Q, Chen Y. Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning. IEEE Syst J. 2023;17(2):2500–11. doi:10.1109/jsyst.2023.3249217.
14. Bai X, Cao M, Yan W, Ge SS, Zhang X. Efficient heuristic algorithms for single-vehicle task planning with precedence constraints. IEEE Trans Cybern. 2020;51(12):6274–83. doi:10.1109/tcyb.2020.2974832.
15. Zhu S, Song Z, Huang C, Zhu H, Qiao R. Dependency-aware cache optimization and offloading strategies for intelligent transportation systems. J Supercomput. 2025;81(1):45. doi:10.1007/s11227-024-06596-7.
16. Hussain MM, Azar AT, Ahmed R, Umar Amin S, Qureshi B, Dinesh Reddy V, et al. SONG: a multi-objective evolutionary algorithm for delay and energy aware facility location in vehicular fog networks. Sensors. 2023;23(2):667. doi:10.3390/s23020667.
17. Materwala H, Ismail L, Shubair RM, Buyya R. Energy-SLA-aware genetic algorithm for edge-cloud integrated computation offloading in vehicular networks. Future Gen Comput Syst. 2022;135:205–22. doi:10.1016/j.future.2022.04.009.
18. Sun Y, Wu Z, Shi D, Hu X. Task offloading method of internet of vehicles based on cloud-edge computing. In: 2022 IEEE International Conference on Services Computing (SCC); 2022 Jul 11–16; Barcelona, Spain. p. 315–20.
19. Wu X, Dong S, Hu J, Hu Q. Multi-objective computation offloading based on invasive tumor growth optimization for collaborative edge-cloud computing. Soft Comput. 2023;27(23):17747–61. doi:10.1007/s00500-023-09051-6.
20. Cui Z, Xue Z, Fan T, Cai X, Zhang W. A many-objective evolutionary algorithm based on constraints for collaborative computation offloading. Swarm Evol Comput. 2023;77:101244. doi:10.1016/j.swevo.2023.101244.
21. Moghaddasi K, Rajabi S, Gharehchopogh FS, Ghaffari A. An advanced deep reinforcement learning algorithm for three-layer D2D-edge-cloud computing architecture for efficient task offloading in the Internet of Things. Sustain Comput Inform Syst. 2024;43:100992. doi:10.1016/j.suscom.2024.100992.
22. Lin J, Huang S, Zhang H, Yang X, Zhao P. A deep reinforcement learning based computation offloading with mobile vehicles in vehicular edge computing. IEEE Internet Things J. 2023;10(17):15501–14. doi:10.1109/jiot.2023.3264281.
23. Lu J, Jiang J, Balasubramanian V, Khosravi MR, Xu X. Deep reinforcement learning-based multi-objective edge server placement in Internet of Vehicles. Comput Commun. 2022;187:172–80. doi:10.1016/j.comcom.2022.02.011.
24. Liu L, Yuan X, Zhang N, Chen D, Yu K, Taherkordi A. Joint computation offloading and data caching in multi-access edge computing enabled internet of vehicles. IEEE Trans Veh Technol. 2023;72(11):14939–54. doi:10.1109/tvt.2023.3285073.
25. Geng L, Zhao H, Wang J, Kaushik A, Yuan S, Feng W. Deep-reinforcement-learning-based distributed computation offloading in vehicular edge computing networks. IEEE Internet of Things J. 2023;10(14):12416–33. doi:10.1109/jiot.2023.3247013.
26. Rahmani AM, Haider A, Khoshvaght P, Gharehchopogh FS, Moghaddasi K, Rajabi S, et al. Optimizing task offloading with metaheuristic algorithms across cloud, fog, and edge computing networks: a comprehensive survey and state-of-the-art schemes. Sustain Comput Inform Syst. 2025;45:101080. doi:10.1016/j.suscom.2024.101080.
27. Yao L, Xu X, Bilal M, Wang H. Dynamic edge computation offloading for internet of vehicles with deep reinforcement learning. IEEE Trans Intell Transp Syst. 2023;24(11):12991–9. doi:10.1109/tits.2022.3178759.

28. Liu Z, Jia Z, Pang X. DRL-based hybrid task offloading and resource allocation in vehicular networks. Electronics. 2023;12(21):4392. doi:10.3390/electronics12214392.
29. Qiu B, Wang Y, Xiao H, Zhang Z. Deep reinforcement learning-based adaptive computation offloading and power allocation in vehicular edge computing networks. IEEE Trans Intell Transp Syst. 2024;25(10):13339–49. doi:10.1109/tits.2024.3391831.
30. Shang C, Sun Y, Luo H, Guizani M. Computation offloading and resource allocation in NOMA-MEC: a deep reinforcement learning approach. IEEE Internet of Things J. 2023;10(17):15464–76. doi:10.1109/jiot.2023.3264206.
31. Ullah I, Singh SK, Adhikari D, Khan H, Jiang W, Bai X. Multi-agent reinforcement learning for task allocation in the internet of vehicles: exploring benefits and paving the future. Swarm Evol Comput. 2025;94:101878. doi:10.1016/j.swevo.2025.101878.
32. Yan J, Bi S, Duan L, Zhang YJA. Pricing-driven service caching and task offloading in mobile edge computing. IEEE Trans Wireless Commun. 2021;20(7):4495–512. doi:10.1109/twc.2021.3059692.
33. Ko SW, Kim SJ, Jung H, Choi SW. Computation offloading and service caching for mobile edge computing under personalized service preference. IEEE Trans Wireless Commun. 2022;21(8):6568–83. doi:10.1109/twc.2022.3151131.
34. Zhu S, Tian X, Zhang Z, Qiao R, Zhu H. Content placement and edge collaborative caching scheme based on deep reinforcement learning for internet of vehicles. IEEE Trans Intell Transp Syst. 2025;26(6):8050–64. doi:10.1109/tits.2025.3558898.
35. Tang C, Zhu C, Wu H, Li Q, Rodrigues JJ. Toward response time minimization considering energy consumption in caching-assisted vehicular edge computing. IEEE Internet of Things J. 2021;9(7):5051–64. doi:10.1109/jiot.2021.3108902.
36. Xue Z, Liu Y, Han G, Ayaz F, Sheng Z, Wang Y. Two-layer distributed content caching for infotainment applications in VANETs. IEEE Internet of Things J. 2022;9(3):1696–711. doi:10.1109/jiot.2021.3089280.
37. Chen W, Su Z, Xu Q, Luan TH, Li R. VFC-based cooperative UAV computation task offloading for post-disaster rescue. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications; 2020 Jul 6–9; Online. p. 228–36. doi:10.1109/infocom41043.2020.9155397.
38. Bi S, Huang L, Zhang YJA. Joint optimization of service caching placement and computation offloading in mobile edge computing systems. IEEE Trans Wireless Commun. 2020;19(7):4947–63. doi:10.1109/twc.2020.2988386.
39. Yu S, Chen X, Yang L, Wu D, Bennis M, Zhang J. Intelligent edge: leveraging deep imitation learning for mobile edge computation offloading. IEEE Wireless Commun. 2020;27(1):92–9. doi:10.1109/mwc.001.1900232.
40. Zhang Y, Liu T, Zhu Y, Yang Y. A deep reinforcement learning approach for online computation offloading in mobile edge computing. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS); 2020 Jun 15–17; Hangzhou, China. p. 1–10.
41. Mao Y, Zhang J, Letaief KB. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Selected Areas Commun. 2016;34(12):3590–605. doi:10.1109/jsac.2016.2611964.
42. Xu Z, Liu X, Jiang G, Tang B. A time-efficient data offloading method with privacy preservation for intelligent sensors in edge computing. EURASIP J Wireless Commun Netw. 2019;2019(1):1–12. doi:10.1186/s13638-019-1560-8.
43. Ye Q, Shi W, Qu K, He H, Zhuang W, Shen X. Joint RAN slicing and computation offloading for autonomous vehicular networks: a learning-assisted hierarchical approach. IEEE Open J Veh Technol. 2021;2:272–88. doi:10.1109/ojvt.2021.3089083.
44. Van Moffaert K, Drugan MM, Nowé A. Scalarized multi-objective reinforcement learning: novel design techniques. In: 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL); 2013 Apr 16–19; Singapore. p. 191–9.
45. Brys T, Harutyunyan A, Vrancx P, Nowé A, Taylor ME. Multi-objectivization and ensembles of shapings in reinforcement learning. Neurocomputing. 2017;263:48–59. doi:10.1016/j.neucom.2017.02.096.
46. Gorbachenko VI, Alqezweeni MM. Learning radial basis functions networks in solving boundary value problems. In: 2019 International Russian Automation Conference (RusAutoCon); 2019 Sep 8–14; Sochi, Russia. p. 1–6.

47. Alqezweeni MM, Gorbachenko V. Approximation of functions and approximate solution of partial differential equations using radial basis functions networks. In: 2020 1st. Information Technology To Enhance e-learning and Other Application (IT-ELA); 2020 Jul 12–13; Baghdad, Iraq. p. 25–30. doi:10.1109/it-ela50150.2020.9253069.
48. Li H, Chen C, Shan H, Li P, Chang YC, Song H. Deep deterministic policy gradient-based algorithm for computation offloading in IoV. IEEE Trans Intell Transp Syst. 2024;25(3):2522–33. doi:10.1109/tits.2023.3325267.
49. Wu J, Wang J, Chen Q, Yuan Z, Zhou P, Wang X, et al. Resource allocation for delay-sensitive vehicle-to-multi-edges (V2Es) communications in vehicular networks: a multi-agent deep reinforcement learning approach. IEEE Trans Netw Sci Eng. 2021;8(2):1873–86. doi:10.1109/tnse.2021.3075530.
50. Liu X, Feng L, Zhang P, Yu Y, Wang J. An efficient computational offloading method using deep reinforcement learning in edge-end-cloud. Ad Hoc Netw. 2025;178:103941. doi:10.1016/j.adhoc.2025.103941.
51. Mossalam H, Assael YM, Roijers DM, Whiteson S. Multi-objective deep reinforcement learning. arXiv:1610.02707. 2016.
52. Zhu S, Tian X, Chen H, Zhu H, Qiao R. Edge collaborative caching solution based on improved NSGA II algorithm in Internet of Vehicles. Comput Netw. 2024;244:110307. doi:10.1016/j.comnet.2024.110307.