Journal of Computing and Security

Research Article, 2025, Volume 12, Number 2 (pp. 107–128)

http://www.jcomsec.org

A Q-Learning Approach for Dynamic Resource Management in Three-Tier Vehicular Fog Computing

Bahar Mojtabaei Ranani [a] Mahmood Ahmadi [a,*] Sajad Ahmadian [b]

[a] *Department of Computer Engineering and Information Technology, Razi University, Iran.*

[b] *Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran.*

ARTICLE INFO.

Article history:

Received: 24 October 2025

Revised: 01 January 2026

Accepted: 15 February 2026

Published Online: 20 April 2026

Keywords:

Task Scheduling, Vehicular Fog Computing, Q-learning, Resource Allocation.

ABSTRACT

In this paper, a method for predicting the resources required for an intelligent vehicle client using a three-layer vehicular computing architecture is proposed. This method leverages Q-Learning to optimize resource allocation and enhance overall system performance. This approach employs reinforcement learning capabilities to provide a dynamic and adaptive strategy for resource management in a fog computing environment. The key findings of this study indicate that Q-learning can effectively predict the appropriate allocation of resources by learning from past experiences and making informed decisions. Through continuous training and updating of the Q-learning agent, the system can adapt to changing conditions and make resource allocation decisions based on real-time information. The experimental results demonstrate the effectiveness of the proposed method in optimizing resource allocation. The Q-learning agent predicts the optimal values for memory, bandwidth, and processor. These predictions not only minimize resource consumption but also meet the performance requirements of the fog system. Implementations show that this method improves the average task processing time compared to other methods evaluated in this study.

1 Introduction

Smart transportation and driving have experienced exponential growth in recent years. This is driven by the Internet of Vehicles (IoV), which enhances vehicle connectivity and enables communication with other vehicles and surrounding infrastructure. The IoV has significantly boosted the adoption of other emerging technologies, transforming how we interact with vehicles. For example, autonomous bus fleets in busy environments must interact with other vehicles to prevent accidents and reduce traffic, making real-time decisions without delay. Additionally, in-vehicle entertainment and autonomous driving have revolutionized the driving experience [1]. Undoubtedly, computations such as selecting the optimal route and estimating the distance to the vehicle ahead require resources like the cloud. However, the significant distance between objects and the cloud, along with the high volume of delay-sensitive requests, creates challenges in service delivery. Nevertheless, leveraging the computational capabilities of idle resources near end devices, such as manned or unmanned vehicles, and establishing a fog computing framework known as vehicular fog computing can reduce the number of requests sent to the

* Mahmood Ahmadi.

Email addresses: baharmojtabaei7125@gmail.com (B. Mojtabaei), m.ahmadi@razi.ac.ir (M. Ahmadi), s.ahmadian@kut.ac.ir S. Ahmadian

https://dx.doi.org/10.22108/jcs.2026.147193.1184

ISSN: 2322-4460

cloud and decrease response times. Additionally, the offloading process is challenging, and the limited capacity of uplink connections between edge and cloud servers, due to network congestion issues, cannot support the delivery of large-scale content [2].

Nevertheless, resource constraints in vehicular fog computing, along with the dynamic changes and mobility in this system, lead to significant challenges. These include identifying the optimal resource in terms of computational power and ensuring resource availability within the timeframe from request to response. Consequently, intelligent resource allocation in vehicular fog computing requires methods such as machine learning. These methods learn environmental behavior to some extent based on end-device requests and responses, enabling optimal predictions for allocating resources to new requests.

How can machine learning algorithms be used to optimally predict the resources needed in vehicular fog computing to process a request and provide a response with minimal delay? To achieve the best prediction of required resources in vehicular fog computing, machine learning algorithms such as hierarchical meta-reinforcement learning can be employed. These algorithms enable the processing of requests and delivery of responses with minimal delay. These algorithms can make optimal resource allocation decisions based on input data and the current state of resources. By training these algorithms with input data and required outputs, improvements in resource prediction and reduced response latency can be achieved.

In this paper, a proposed method using Q-learning for managing and predicting the required resources of a vehicle's fog computing system with a three-layer architecture was presented. This method offers a dynamic and adaptive approach for resource management in a computational environment, utilizing the capabilities of reinforcement learning. The three-layer architecture of the vehicular fog computing system is composed of vehicle, fog, and cloud layers, providing a distributed and non-centralized approach to resource management. The vehicle layer uses local resources, while the fog layer acts as an intermediary, offering additional computational and storage capabilities. The cloud layer provides scalable and extensive resources for performing heavier computational tasks. The results of the experiments demonstrate the effectiveness of the proposed method in optimizing resource allocation. The Q- agent learns optimal actions that minimize resource consumption while meeting the system's performance requirements. This leads to an improvement in efficiency, a reduction in delay, and an increase in the reliability of the system. The main contributions of this paper are as follows:

- Q-Learning integration in three-tier VFC: Novel application of Q-Learning for real-time prediction of CPU, memory, and bandwidth in a vehicle-fog-cloud architecture, adapting to dynamic vehicular workloads unlike static baselines.
- Custom adaptive framework: Introduces a state space encompassing traffic conditions, resource utilization, and app demands, with actions enabling long-term reward maximization through error-learning and multi-objective balancing.
- Task probability-based rewards: Pioneers a reward system tied to task arrival probabilities, optimizing allocation to reduce latency and waste, outperforming methods like FCFS and RR in high-mobility scenarios.

The remainder of the paper is organized as follows. Section 2 presents the related research in vehicular fog computing resource management. Section 3 describes the proposed approach for resource prediction in vehicular fog computing systems. Section 4 evaluates the proposed solution. Finally, the Section 5 concludes the paper.

2 Related Works

This section reviews related research in three parts, including: The works based on offloading computation to vehicle fog computing nodes, the works based on machine learning in vehicular fog computing, and the works based on predicting the resource requirements in vehicular fog computing.

2.1 Based on Offloading Computation to Vehicle Fog Computing Nodes

To incentivize vehicles for task acceptance in fog computing, mechanisms like base station contracts are crucial. These contracts define the relationship between efficiency (computational resources) and reward (payment to vehicles). Vehicles select contracts to maximize profit and act as fog nodes. Task allocation is modeled as a two-sided matching game, considering delay, task size, and cost [3]

In [4], machine learning is used for task allocation to nodes in vehicular fog computing. A roadside processing unit (RSU), having learned system performance, advises other units and vehicles. The learning agent interacts with fog nodes to gain performance knowledge. After acquiring complete knowledge, tasks are assigned to nodes with the best performance.

The FOLO method [5] is proposed to optimize service latency and reduce quality degradation by allocating tasks to fixed and mobile fog nodes. Formulated as a dual-objective optimization problem (minimizing latency and quality loss), the process includes four stages: discovering mobile fog nodes, sending task requests, allocating tasks to fog nodes, and scheduling task transfers. Fixed fog nodes reside in static infrastructure, while mobile nodes are carried by vehicles

with communication modules. The core module manages task allocation and scheduling by tracking node paths and positions.

This approach [6] aims to maximize throughput and reduce queue latency using Lyapunov optimization to separate resource allocation and task offloading. An asynchronous federated deep Q-learning algorithm offloads tasks from user vehicles to edge or fog servers. Collaborative vehicular networks, integrating edge and fog computing, enable real-time data processing. Simulations show improved throughput and effective reduction in end-to-end queue latency.

In vehicular fog computing [7], optimizing the connection between roadside processing units and fog servers is key to reducing energy consumption and balancing computational load. The FedDOVe method, using federated deep Q-learning, reduces energy use by up to 49% and improves load balancing by up to 65%. It leverages global information without requiring a centralized model. The federated system ensures data privacy while enabling collaborative model development across entities.

Limited computational capacity and power supply hinder long-term vehicle participation in vehicular cloud edge computing networks. In [8], an integrated model optimizes energy consumption and latency in task offloading and resource allocation, formulated as a binary optimization problem. A distributed deep learning approach using parallel neural networks provides near-optimal task offloading decisions. Implementation results demonstrate reduced energy consumption.

2.2 Based on Machine Learning in Vehicular Fog Computing

In [9], to allocate resources in vehicular fog computing, supervised learning and historical data are used to minimize service latency. The method assigns service requests locally or to a remote cloud server for optimal latency. In dynamic urban environments, a reinforcement learning algorithm with proximal policy optimization detects changes and allocates resources. Roadside units advertise fog services via WAVE messages, allowing new vehicles to join. Fog resource management periodically updates allocations based on demand and status. Evaluations show this approach outperforms conventional resource allocation schemes in service satisfaction.

In [10], deep reinforcement learning with an incentive contract reduces complexity in resource management and allocation. Tasks are rapidly assigned to vehicles with idle resources via deep neural networks. A queue model based on accumulated rewards prevents conflicts from simultaneous task assignments. System performance gradually improves with more vehicles, and mini-batch processing enhances efficiency. Increasing vehicle numbers boosts roadside processing tasks, though idle computational resources also rise. Mini-batch processing reduces computational load and improves model generalization.

This study [11] proposes two schemes for joint task assignment and resource allocation in vehicular fog computing. The first scheme divides the problem into two subproblems—task and resource allocation—solved using matching and convex optimization. The second scheme enhances accuracy using 0-1 integer linear programming. Both schemes leverage vehicle mobility predictions to forecast network topology, optimizing task and resource allocation. The goal is to minimize overall latency in vehicular fog computing with a TARA strategy. Simulations show these schemes effectively reduce total latency.

In [12], parked vehicles can serve as fog nodes to process mobile app demands, enhancing wireless network communication. The system comprises three components: a fog controller, parked electric vehicles, and mobile devices. The fog controller manages computational requests, directing them to vehicles or the cloud based on available energy. Requests follow a Poisson process, and vehicle exits follow an exponential distribution. Charge optimization is formulated using a Markov decision process to improve app responsiveness. This approach enhances rewards and battery lifespan through equitable energy allocation.

Vehicular edge computing supports emerging applications in vehicular networks using pervasive edge resources. High vehicle mobility and dynamic network topology complicate task scheduling and resource allocation [13]. A matching-based algorithm is proposed to optimize task offloading and resource allocation, reducing latency and costs. It addresses mobility, offloading decisions, resource allocation, and reliable result feedback. Simulations show superior performance compared to existing strategies. The goal is to efficiently utilize resources for low-latency, reliable services.

The paper [14] explores vehicle edge computing for automotive services, proposing a framework for intelligent management of heterogeneous resources across vehicles, edge servers, and the cloud. It supports vehicle-to-vehicle and vehicle-to-infrastructure offloading. Resource allocation and task scheduling are jointly optimized to enhance system efficiency and meet user needs. An asynchronous actor-critic algorithm is used for optimal scheduling decisions. The framework aims to maximize resource utilization and improve user service experience. It integrates inter-layer (local, edge, cloud) and intra-layer (among edge servers or users) collaboration

2.3 Based on Predicting the Resource Requirements in Vehicular Fog Computing

In [15], by formulating the resource allocation problem, one of the advanced reinforcement learning methods, Actor-Critic, was used to predict the availability of resources in vehicular fog computing. First, the resource allocation problem in vehicular fog computing is formulated to maximize user satisfaction. Then, a set of all divided regions and the collection of vehicular services are presented. Additionally, user satisfaction is defined in terms of service latency and memory space. The proposed Q-Learning method, a value-based reinforcement learning approach, differs from Actor-Critic algorithms by directly estimating the action-value function (Q-values) to select optimal actions in discrete state-action spaces. This makes it off-policy and tabular, enabling straightforward implementation without requiring separate policy and value estimators. In contrast, Actor-Critic combines an actor (policy network) for action selection with a critic (value network) for evaluation, which supports on-policy learning and handles continuous action spaces more effectively but introduces higher computational complexity and potential instability from policy gradient variance. The key advantage of Q-Learning lies in its simplicity, faster convergence in environments like the three-tier vehicular fog computing architecture described, and lower overhead for real-time adaptation. In such settings, resource allocation (e.g., memory, bandwidth, processor) can be discretized effectively, as evidenced by superior task processing times and service rates in our simulations. In contrast, Actor-Critic methods may excel in scalability for larger, continuous domains, though at the cost of increased training demands.

The TARA algorithm [16] is proposed for predicting resource needs, task allocation to vehicles, and resource assignment in wireless vehicle networks. When a vehicle generates a task difficult to process alone, it sends a request to roadside units for optimized task and resource allocation to minimize delay. The task is divided into N subtasks, each processed by a vehicle with varying file and computational sizes. Multi-hop communication extends the vehicular fog computing service range. The proposed scheme significantly improves performance with larger file sizes compared to others and optimizes resource allocation by reducing bandwidth per link. Simulations confirm that this strategy effectively reduces delay.

The paper [17] proposes a deep learning approach for predicting future resource needs. It introduces a resource prediction module and an optimal decision-making module to enhance AI-based resource scheduling. The prediction module is trained with the current request and predicted resource data. The decision-making module integrates prediction metrics and HPA to compute optimal container numbers for the API server. HPA in Kubernetes enables automatic pod scaling based on demand. Results show significant reductions in resource occupancy and response latency (over 25%) in vehicular fog computing.

Resource predicting is essential for fair and efficient allocation. In paper [18], a fog-enhanced vehicle service model is used where vehicles are end-users requesting resources. A probabilistic model predicts fluctuating resource needs accurately. The model categorizes users as privileged and non-privileged. Privileged users receive more resources, with a base price parameter fixed by vehicle type. This approach optimizes resource utilization.

The paper [19] models resource prediction in the Internet of Vehicles using a semi-Markov decision process. It introduces a resource reservation strategy and secondary allocation mechanism for adaptive, self-learning dynamic resource allocation. Suitable for high-mobility vehicular environments, it handles service requests at any time. A joint optimal allocation scheme for communication and edge computing resources enhances long-term system revenue. It accounts for variable wireless channel characteristics to improve transmission efficiency and user experience quality. Simulations show that the mechanism reduces service request rejection rates and improves system performance. Table 1 summarizes the works done in vehicular fog computing resource allocation.

3 Proposed Method

Achieving an efficient resource prediction solution is a challenging task in resource management. Ideally, this solution should, first, avoid resource wastage and, second, ensure that resources are fully sufficient for task allocation. Such requirements are particularly demanding in environments with time-varying workloads. The use of reinforcement learning in the proposed method for predicting required resources in computing (such as CPU allocation, memory, and bandwidth) is driven by several key reasons.

- The dynamic and uncertain nature of systems: Computational environments are typically dynamic and uncertain. Resource demands can continuously change, and traditional algorithms may not optimally cope with these variations. Reinforcement learning enables an agent to learn and gradually improve resource management policies through direct interaction with the environment. This is particularly significant in cloud systems and large-scale infrastructures.
- Ability to adapt to environmental changes: Reinforcement learning enables agents to improve optimal resource allocation policies over time by accumulating experiences. As conditions

Table 1. Summary of Papers on Task and Resource Allocation.

Source	Proposed method	Advantages	Disadvantages	Delay reduction
[3]	It proposes an efficient incentive mechanism based on theoretical contract modeling and transforms the task allocation problem into a two-sided matching problem between vehicles and user equipment (UEs).	Reduced network latency, Optimal performance, and Lower complexity	With a low number of vehicles, delay increases, and price increases as a solution to competition.	∼ 15-20 (inferred from low vehicle count scenarios)
[5]	This paper presents a novel solution for delay and QoS-optimized task allocation in vehicular edge computing, called the FOLO algorithm. FOLO is designed to support the mobility of vehicles.	It reduces service delay by up to 27% and improves overall QoS by up to 56%.	Relatively high computational complexity. With the selection of complex and random nodes, performance decreases.	27
[9]	This paper formulates the problem of fog resource allocation using parked vehicles and integrates a proposed algorithm with reinforcement learning to make more efficient resource allocation decisions.	Improving service latency from the users' perspective and minimizing response time for users.	Additional delay, as well as the computational load of tasks and returning results, can be prone to errors from a data delivery perspective.	∼ 20 (service latency reduction)
[20]	This paper investigates the problem of energy-aware task allocation in vehicular fog networks using reinforcement learning combined with heuristic information, referred to as URLLC.	It reduces interference between nodes, increases throughput, and ensures the reliability of URLLC V2X communications by selecting the best loading decisions.	Sometimes assumptions can be unrealistic while making specific loading decisions that may not yield the best results.	N/A
[12]	In this article, the resource allocation problem is formulated as a Markov Decision Process (MDP), and dynamic programming is used to solve the decision-making problem.	Improving communications in a public wireless network, it performs better than other fixed strategies in terms of reward and energy efficiency.	Continuous use of the same batteries may significantly reduce the lifespan of electric vehicle batteries.	N/A
[10]	This paper proposes a contract theory-based incentive mechanism for the current IoV scenario to encourage vehicles to share their idle computing resources. It also uses a combination of contract-based DRL deep reinforcement learning.	Encourage vehicles to share their idle resources. Reduce the complexity of implementing resource management and allocation.	System performance improves later.	N/A
[15]	This paper formulates the vehicular fog computing resource allocation problem and uses reinforcement learning to predict the availability of vehicular fog computing resources and service demand.	Maximizes user satisfaction.	Considering the high mobility of vehicles in vehicular fog computing and the rapid change of network topology, the problem of vehicle coordination for vehicular fog computing is a big challenge.	N/A
[6]	This paper proposes a partial V2V offloading scheme to solve the high mobility of vehicles, which measures the availability of vehicle services based on the duration of the connection and the diversity of its computational resources. It also uses a DRL algorithm based on SAC software.	Maximizing the expected reward over a period of time and performing better than other methods.	The highly dynamic vehicle environment that affects the vehicle's task offloading performance does not consider infrastructure as fog nodes.	N/A
[21]	In this paper, reducing service time by allocating available bandwidth to four types of services is formulated, and network tools are used as a measurement index for allocating available bandwidth.	Accelerate service delivery, save more energy, and improve system survivability.	Power supply limitations of mobile computing devices.	High (∼ 30-40, accelerates delivery)
[22]	This paper presents EnTruVe, an energy-aware virtual machine allocation and TRUst in vehicle fog computing solution that aims to minimize the number of virtual machine migrations and reduce the energy consumption associated with virtual machine processing as much as possible.	Minimize the number of virtual machine migrations and reduce the energy consumption associated with virtual machine processing. The most energy-efficient solution.	The number of virtual machine migrations between fog nodes can reduce overall system performance.	N/A
[16]	This paper, considering the mobility effect in vehicle fog computing, studies the joint task allocation and resource allocation optimization problem, and formulates the joint optimization problem from the Max-Min perspective to reduce the overall task delay.	Minimizing the delay, and also the proposed scheme reduces the bandwidth allocated to each link by keeping the total bandwidth constant and increasing the number of vehicles.	This scheme does not provide significant performance improvements when the file size is small.	High (∼ 40, min delay)
[13]	The problem of shared task loading and resource allocation in vehicular networks is studied to optimize the system tools to reduce the latency and cost of computing and communication services. To solve this problem, a matching-based task loading and resource allocation algorithm is proposed.	It minimizes the latency and cost of using heterogeneous communications and computing resources located in roadside processing units and vehicles.	Does not consider security and privacy.	High (∼ 35, min latency/cost)
[14]	Joint workload computing and demand-based resource allocation in automotive edge computing for automotive services and applications are investigated.	Maximizes resource utilization. Ensures a diverse user experience in vehicle edge computing.	The number of vehicles is limited, and resource allocation may be delayed as the number of vehicles increases.	N/A
[23]	To review and present a new algorithm for allocation and optimization of exploration resources in a MultiFog-Cloud environment.	Optimize runtime and communication latency. Minimize cloud load.	Does not optimize revenue. Does not consider mobile users' preferences.	Medium (∼ 20, runtime/latency)
Proposed approach	Q-Learning-based reinforcement learning in a three-tier (vehicle-fog-cloud) architecture for real-time prediction of CPU, memory, and bandwidth.	Adaptive to dynamic vehicular mobility and workloads, learning optimal allocations from experience without predefined rules.	Q-learning growth in large state-action spaces may increase memory overhead on resource-constrained fog nodes. Sensitive to reward weighting; imbalances could prioritize one objective	∼ 35-45 (inferred from APT graphs)

change, such as varying processing loads or increasing/decreasing resource demands, the agent can dynamically adapt.

- Long-term optimization: In reinforcement learning, the goal is to enable the agent to maximize long-term rewards rather than make short-term decisions. In resource management, a decision that appears optimal in the short term may lead

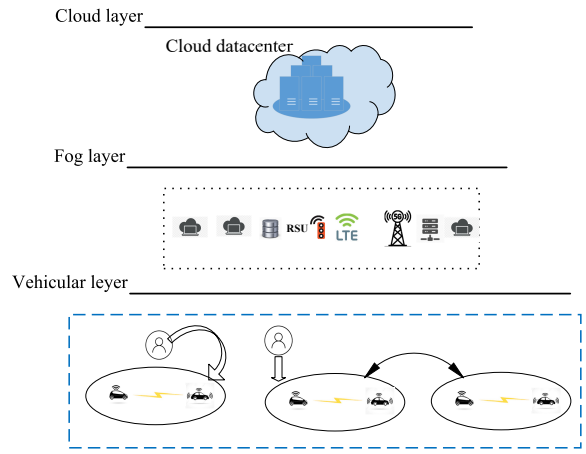
to issues in the long term (e.g., resource overloading). Reinforcement learning can strike a balance and make decisions that are more optimal in the long run.

- Addressing complexity and high scalability: Managing resources in large and complex systems requires algorithms capable of optimizing in the face of vast amounts of data and diverse states. Reinforcement learning can handle high complexity and large state spaces (e.g., various combinations of workloads and resource demands) more effectively than traditional methods.
- Learning from errors: Reinforcement learning allows agents to learn from their errors and adjust their policies accordingly. In resource allocation problems, this can lead to continuous improvement of system performance and prevent the repetition of previous errors.
- Solving multi-objective problems: In many cases, optimal resource allocation must simultaneously consider multiple criteria, such as reducing latency, minimizing energy consumption, or lowering economic costs. Reinforcement learning can optimize these multi-objective problems concurrently. As a result, due to its adaptability, long-term optimization capabilities, and dynamic learning, reinforcement learning is a suitable choice for the problem of prediction and resource allocation in computational environments.

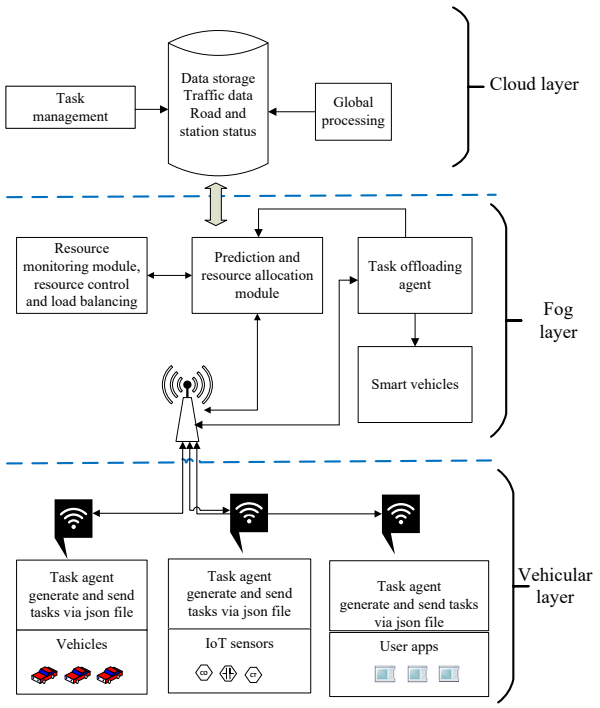
Figure 1(a) shows the implementation of a vehicle fog computing architecture. The figure depicts the various components of the system and how they interact. According to Figure 1(a), the vehicular fog computing architecture consists of the following layers: vehicle layer, fog layer, and cloud layer. The vehicle layer represents the vehicles in the network, the fog layer includes local fog nodes, and the cloud layer represents centralized cloud data centers. The design and implementation of the network is based on appropriate settings and connections between layers. To design and implement the vehicular fog computing architecture, each layer is considered separately, and the settings and connections of the layers are defined. It should be noted that the arrows between the layers indicate the assignment of the task to a higher layer for processing.

Vehicle Layer: The vehicle layer represents the vehicles in the network. Each vehicle will be equipped with communication and computing capabilities. The configuration of the vehicle layer tasks includes the following:

- Communication technology: Vehicles may use technologies such as Wi-Fi or cellular networks to communicate.
- Computing capabilities: Each vehicle will have built-in computing resources to process and analyze data.



(a) A General VFC Architecture.



(b) A VFC Implementation.

Figure 1. VFC: (a) General Architecture; (b) Implementation.

- Vehicle-to-vehicle communication (vehicular communications): Vehicles must be able to communicate with each other to share information and coordinate their actions.
- Vehicle-to-infrastructure communication: Vehicles must communicate with roadside infrastructure, such as traffic lights or road sensors, to collect data in real time.

Fog layer: The fog layer consists of local fog nodes located in the vicinity of vehicles. These nodes provide additional computing and storage resources to support latency-sensitive and real-time applications. The fog layer configuration includes the following:

- Fog node location: Fog nodes should be strategically placed to ensure coverage and minimize latency. They can be placed in areas with high vehicle density or critical infrastructure points.
- Connectivity: Fog nodes should be connected to the vehicle layer via reliable, low-latency communication links such as wired connections or high-speed wireless networks.
- Computing and storage resources: Fog nodes must have sufficient processing power and storage capacity to handle the computing needs and deployed applications.
- Load balancing: Multiple fog nodes can be deployed to distribute the computing load and provide fault tolerance.

Cloud layer: The cloud layer represents centralized cloud data centers that provide global-scale computing and storage capabilities. The cloud layer configuration includes the following:

- Cloud data center location: Cloud data centers should be strategically located to provide efficient access to fog nodes and minimize network latency.
- Connectivity: Cloud data centers should be connected to the fog layer using high-capacity network links, such as fiber optic connections, to handle the large volume of data generated by vehicles.
- Compute and storage resources: Cloud data centers must have powerful compute infrastructure and large-scale storage facilities to support intensive applications and perform complex data analytics.
- Scalability and elasticity: Cloud data centers must be scalable to handle increasing demand and provide elasticity to dynamically allocate or release compute resources based on workload.

Figure 1(b) shows the implementation aspects of a resource allocation and prediction system using a reinforcement learning algorithm, which includes various components for task management and data processing. The following describes each component and its interactions:

- Vehicular layer: This layer includes users, objects, and applications that generate tasks and send each task to the roadside units in the form of a JSON file. The content of the JSON file includes data fields such as resources and their number, executable code, parameters sent, parameters received, etc. Below are a few examples of entities in the customer layer, briefly described:

(1) Vehicles: These vehicles generate tasks according to their needs. Smart cars may interact with IoT sensors to collect environmental data and make better decisions based on it.

(2) IoT Sensors: IoT sensors collect data related to the environment, vehicle status, etc., and send the relevant tasks to the system. This data can include parameters such as speed, geographical location, temperature, humidity, etc. After collection, this data is sent to the computing layer at the edge. This communication is usually done through wireless networks. In the computing edge layer, the data received from the sensors is processed. This processing includes initial analysis, data filtering, and immediate decision-making that does not require sending the data to the cloud. After initial processing, the information can be used to assign tasks to different nodes.

(3) User applications: Applications act as the user interface in distributed computing systems and play a key role in data collection and task generation. These programs collect information from users and their surroundings, such as location and specific needs, and based on that, create tasks for processing. After sending these tasks to the edge and cloud computing layers, the processed results are returned to the applications for display to users. This improves the user experience and provides more accurate information.

- Fog layer: This layer deals with local processing, task management, node resource management, and task resource prediction, and includes thefollowing modules:

(1) Task offloading agent: This module receives tasks from the client layer and manages them. First, the agent in this module collects and analyzes information about the resource status of the nodes, their current workload, and their capacity. Then, based on this analysis, the resource prediction and allocation module estimates the amount of resources required and decides which node is the most suitable option for processing the task. After selecting a node, the agent is responsible for transferring the tasks to the destination nodes and monitoring their execution performance. It also calls the resource monitoring module to update the resource table for those nodes. This process helps optimize system performance and ensure efficient resource distribution. Finally, it returns feedback on the waiting time and execution time of the tasks to the predicting and allocation module for optimization.

(2) Smart cars with processing resources: These cars act as server computing nodes in the environment and are responsible for executing tasks or sending them to other resources for analysis. Smart cars perform a variety of tasks and act as executive units in the system. These cars are able to collect data and provide feedback to the

system. After tasks are assigned in the computing layer, information about how to perform these tasks and the resources required is sent to the smart cars so that they can effectively perform their tasks.

(3) Resource prediction and allocation module: This module employs the Q-Learning algorithm and data analysis techniques to examine feedback from executed tasks, workload patterns, and resource consumption trends. The aim is to generate accurate predictions of future task requirements. After analyzing the data, this module allocates resources optimally to tasks and decides how many resources to consider for executing tasks to maximize system efficiency. This process gradually reduces latency, increases efficiency, and improves workload between different nodes in the system.

(4) Resource monitoring module: This module balances the workload of VFC layer nodes by monitoring the amount of resources used and free for each node. Resource monitoring is a key tool for effective management of computing systems. This process involves tracking and recording the real-time usage of each node's processor, memory, bandwidth, and storage to prevent problems such as overloading and insufficient resources. By analyzing the collected data, performance optimization and resource reallocation can be done, and also using historical data to predict future needs.

- Cloud layer: This layer handles big data processing and storage of related information:

(1) Task management: A central location for storing information about traffic, road conditions, and stations, and for receiving management and scheduling.

(2) Global processing: Analyzing data on a large scale and performing complex calculations.

3.1 Problem Modeling

This section defines the variables and factors that are useful in the resource allocation decision-making process. These factors include the utilization of available resources, traffic conditions, and application needs. The model features a state space comprising a set of states and a set of actions for each state. When an agent acts, it transitions from one state to another. Executing an action in a particular state yields a reward (a numerical score) for the agent.

3.1.1 State Space

A state space is a multidimensional space in which each dimension represents one of the variables. The values within each dimension can be discrete or continuous, depending on the nature of the variable. Usually, one variable is not sufficient for each of the factors of traffic conditions, current resource usage, and resource availability. Usually, multiple variables are needed to accurately represent and capture the complexity of these factors. For example, traffic conditions can include variables such as the number of concurrent requests, the type of applications, and the expected demand for each type. Similarly, current resource utilization may consist of multiple variables such as CPU utilization, memory utilization, and network bandwidth utilization. Resource availability can also include variables such as the number of available compute nodes, the amount of free disk space, and network capacity. By considering multiple variables for each factor, a state space can provide a more comprehensive representation of the system state for decision-making. Some of these important variables and factors that may play a role in resource allocation decisions include:

- Current resource utilization: This refers to the current usage levels of various resources such as CPU, memory, disk space, or network bandwidth. It provides information about the amount of resources currently being consumed by the system.
- Traffic conditions: This variable considers the current workload of the system, such as the number of requests, application type, and expected demand. This helps in adjusting the resource allocation based on the current system load.
- Application requirements: These are the specific needs or demands of the application running on the system. This includes factors such as response time, service level agreements, throughput requirements, or any other application-specific criteria.
- Resource availability: This factor reflects the availability of resources in the system, taking into account capacity and any potential constraints. This includes information about the number of available compute nodes, storage devices, or any other resources in the infrastructure.

Table 2 depicts the state space variables and their values.

The total number of variables will be the sum of the variables in each category, which includes current resource usage, traffic conditions, application requirements, and resource availability. In this case, there are 12 variables. For example, if CPU usage, memory usage, and disk space usage are considered as relevant variables, a three-dimensional state space is defined.

$$S = \{(X_0, X_1, X_2) \mid 0 \le X_0 \le M_1, \; 0 \le X_1 \le M_2, \; 0 \le X_2 \le M_3\} \quad (1)$$

In Eq. 1, S represents the state space, $X0$, $X1$, and $X2$ represent the CPU utilization, memory utilization, and disk space utilization, respectively. $M1$, $M2$, $M3$

Table 2. System Variables and Their Values

Current Resource Usage		
Variable	Meaning	Value
CU	CPU usage	{Low, medium, high}
MU	Memory usage	{Low, medium, high}
DSU	Disk space usage	{Low, medium, high}
NBU	Network bandwidth usage	{Low, medium, high}
Traffic Conditions		
Variable	Meaning	Value
NR	Number of requests	{Low, medium, high}
AT	Application type	{Light, medium, heavy}
ED	Expected demand	{Low, medium, high}
Application Requirements		
Variable	Meaning	Value
RT	Response time	{Fast, medium, slow}
SLA	Service level agreement	{Fulfilled, not fulfilled}
OR	Operational requirements	{Low, medium, high}
Resource Availability		
Variable	Meaning	Value
NCN	Number of computation node	{Low, medium, high}
ASD	Availability of storage devices	{Low, medium, high}

are the maximum utilization values for each resource. This state space can be used to evaluate different resource allocation decisions and find the optimal allocation based on the current state of the system. By analyzing different points in the state space, the best resource utilization can be determined by considering other factors.

3.1.2 Action Space

This section specifies the actions that can be performed in the resource allocation process. These actions can include increasing or decreasing resources, transferring computational tasks, or offloading tasks between layers. Action space refers to the set of all available operations that can be performed in resource allocation. In the context of resource allocation in computing systems, the action space includes various operations such as increasing or decreasing resources, transferring computational tasks, and offloading tasks between different layers or components.

3.1.3 Reward Function

A reward function is designed that reflects the goals and objectives of the system. This could include minimizing resource waste, maximizing resource utilization, reducing response time, or maintaining a desired quality of service. The reward function should guide the reinforcement learning process toward desired behaviors and outcomes. The reward function in reinforcement learning minimizes the immediate desirability of a particular state or action taken by an agent in an environment. This is crucial in guiding the agent's learning process and incentivizing it to achieve the system's goals and objectives. The specific design of the reward function depends on the nature of the problem and the desired behaviors and outcomes.

The reward function is specified based on the defined goals. The goals are minimizing resource wastage, maximizing resource utilization, reducing response time, and maintaining quality of service. Therefore, the reward function can be defined based on Eq. 2. The value of the weights w_1, w_2, w_3, and w_4 are determined in trial and error manner and are set to 0.3, 0.3, 0.2, and 0.2.

$$\begin{aligned} r(s,a) = &-w_1 \cdot \text{resource}_{\text{wastage}}(s,a) \\ &+ w_2 \cdot \text{resource}_{\text{utilization}}(s,a) \\ &+ w_3 \cdot \text{response-time}(s,a) \\ &+ w_4 \cdot \text{qos}(s,a) \end{aligned} \tag{2}$$

In Eq.2, the first term of reward is calculated as the negative product of the weighting coefficient $w1$ and the amount of resource waste. By penalizing wasteful actions, the agent is encouraged to minimize resource use. The weighting factor w_1 is determined when implementing a reinforcement learning system. This indicates the importance or priority given to minimizing resource waste. The implementation is chosen based on their understanding of the system and their desired trade-offs between resource use and wasting some amount of w_1. A higher value of w_1 places more emphasis on minimizing waste, while a lower value gives more priority to other aspects of the system's goals. The specific value of w_1 can be adjusted through trial and error to find the optimal balance for the system. The coefficient w_2 is the scaling factor k and is determined through trial and error during implementation. Here, the goals that are achieved by the reward function are:

(1) Minimizing resource waste:

$$\begin{aligned} \text{resource}_{\text{wastage}}(s,a) = \Bigg[&\sum_{i=0}^{n} (A_C - E_C) \\ &+ \sum_{i=0}^{n} (A_M - E_M) \\ &+ \sum_{i=0}^{n} (A_B - E_B) \Bigg] \end{aligned} \tag{3}$$

- A_C: Actual CPU usage
- E_C: Efficient CPU usage
- A_M: Actual memory usage

- E_M: Efficient memory usage
- A_B: Actual bandwidth usage
- E_B: Efficient bandwidth usage

Actual CPU utilization refers to the current level of CPU utilization in a system, indicating how much processing power is being used at any given time. It represents the real-time demand for CPU resources by running processes or programs. To calculate resource wastage, it can be determined by measuring the difference between the actual resources used and the optimal or expected resources required for a given task or operation. This can involve comparing factors such as CPU utilization, memory utilization, network bandwidth, and power consumption against thresholds. For example, if a fog system is responsible for processing a set of tasks, resource wastage can be calculated by summing the differences between the actual CPU utilization for each task and the CPU utilization that was sufficient to efficiently complete the tasks. The same approach can be applied to other resources, such as memory or network bandwidth.

(2) Maximize resource utilization: When considering different types of quantities, such as CPU usage, memory usage, and network bandwidth usage, it may not make sense to simply summarize them in a resource usage equation. This is because these quantities may have different units of measurement and scales, which can lead to inaccuracies in the overall calculation. Instead, a more appropriate approach is to normalize each of these quantities individually before multiplying them by the weight. Normalization involves scaling the values of each quantity to a standard range, usually between 0 and 1, so that they can be directly compared and combined in a meaningful way. The equation for normalizing resources with weighting factors can be expressed in Eq. 4.

$$resource_{Utilization}(s,a) = \left(w_{2,1} \cdot \sum_{i=0}^{n} NCU + w_{2,2} \cdot \sum_{i=0}^{n} NMU + w_{2,3} \cdot \sum_{i=0}^{n} NNBU \right) \quad (4)$$

- NCU=normalize(CPU usage)
- NMU=normalize(memory usage)
- NNBU=normalzie(network bandwidth usage)

Where, $w_{2,1}$, $w_{2,2}$, $w_{2,3}$ are the weights assigned to each normalized quantity, and are set to 0.4, 0.3, and 0.3, respectively. The $normalize()$ is a function that scales the values of each quantity to a standard range (e.g., between 0 and 1).

The goals of minimizing resource waste and maximizing resource utilization are not the same. Minimizing resource waste focuses on reducing the amount of resources that are underused or not used. The goal is to optimize resource allocation and prevent unnecessary waste, which can help conserve resources and reduce costs.

(3) Reduce response time: Here, the reward response-time is inversely proportional to the response time. $T_{current}$ represents the current response time and T_{Max} is the maximum desired response time. The closer the response time is to the desired limit, the higher the reward, encouraging the agent to reduce the response time.

$$response - time(s,a) = \frac{T_{max} - T_{current}}{T_{max}} \quad (5)$$

According to Eq. 5, as the response time $T_{current}$ approaches the maximum desired response time T_{max}, the amount of reward increases. This encourages the agent to reduce the response time even further to receive more reward; therefore, the higher the reward, the closer the response time is to the desired limit.

(4) Maintaining the desired quality of services: The qos reward is calculated based on the difference between the achieved quality and the desired quality of the service. The coefficient $\frac{1}{e^*}$ is a negative exponential function used to assign higher rewards for achieving quality close to the desired level. This guides the agent to maintain the desired quality throughout his decision-making process.

$$qos(s,a) = \frac{1}{e^{(quality_{desired} - quality)}} \quad (6)$$

The function $e^{-(*)}$, is actually a reduction factor that is used to reduce the maximum reward value. This reduction can be used to adjust the reward value to a desired value or to balance the problem. Quality is measured based on factors such as latency, throughput, and reliability.

In this case, quality can be calculated as the weighted average of these factors using appropriate weights:

$$quality = w_{3,1} \cdot \frac{1}{latency} + w_{3,2} \cdot throughput + w_{3,3} \cdot reliability \quad (7)$$

$w_{3,1}$, $w_{3,2}$, and $w_{3,3}$ are weights assigned to each factor that reflect their importance in de-

termining overall quality. These weights can be determined based on specific system priorities.

It is a measurement of the number of units of information that can be processed in a given time, usually measured in bits per second. It indicates the rate at which data can be successfully transmitted or processed in a system. The parameter reliability refers to the percentage of requests that are successfully processed without encountering failure errors. This indicates the reliability of the system in achieving the expected results.

The four objective functions do not add up directly. Instead, they represent different aspects and goals of the system. Each objective function focuses on a specific goal and encourages the agent to optimize its decisions for the base.

The first objective function, 'minimize resource waste', penalizes wasteful actions and encourages the agent to minimize resource usage. Its goal is to optimize resource allocation and prevent unnecessary waste, ultimately conserving resources and reducing costs.

The second objective function, 'maximize resource utilization', encourages the agent to fully utilize available resources to maximize efficiency. It prioritizes efficient resource utilization, thus maximizing system performance or throughput.

The third objective function, 'reduce response time, ' rewards the agent for reducing response time. This function encourages the agent to prioritize actions that lead to faster processing and improved system responsiveness.

The fourth objective function, 'maintaining the desired quality of service,' aims to incentivize the agent to maintain the desired level of quality in terms of latency, throughput, and reliability. It allocates higher rewards for achieving quality close to the desired level.

The impact of these objective functions on the system is not the same, because each function emphasizes a different aspect and goal. For example, minimizing resource waste and maximizing resource utilization may have contradictory effects. Minimizing resource waste may lead to underutilization of resources, while maximizing resource utilization may lead to overutilization of resources; therefore, finding the optimal balance between these goals is very important.

3.1.4 QL Reinforcement Learning Agent Training

The proposed model is run on all nodes of the fog layer. The collected training data, defined state space, action space, and reward function are used to train the QL reinforcement learning agent. One of the most popular reinforcement learning algorithms is the QL reinforcement learning algorithm, which uses experience to learn to update the value of actions in a given situation, known as the Q-value.

The QL reinforcement learning algorithm is specifically used for Markov reinforcement control problems where the agent operates in a well-defined environment, and its goal is to learn an optimal strategy that maximizes the reward. In this algorithm, the Q-value for each state-action pair is updated as feedback from past experiences. This update is usually done using a QL reinforcement learning update equation.

The algorithm and pseudocode of the training process are given in Algorithm 1.

This pseudocode contains separate functions for each operation, including initializing Q-values, selecting the action using the epsilon-greedy strategy, executing the action in the environment, and updating the Q-value for the current state-action pair. First, the Q-Values for each pair of states and actions are randomly initialized with a predetermined initial value. Then, in the second step, for each part, the following steps are performed: Displaying the current state of the system, looping over time steps, selecting an action using an exploration and exploitation strategy, executing the selected action in the environment, observing the next state and reward, updating the Q value for the current state and action pair using the Q-learning update rule, and moving to the next state.

In the third step, the functions used in this pseudocode, such as *initialize_q_values*() and *update_q_value*(), must be implemented separately to perform the corresponding operations.

The training algorithm using Q-learning is as follows: Input: Number of training iterations (e.g., 1000 iterations), learning coefficient (e.g., 0 and 1), exploration parameter (e.g., 0 and 3), number of states and actions, and Q-value update equation. Output: Q-value table for each state and action pair.

Lines 1 to 3: Define the function *initialize_q_values*, which returns a zero-filled array of dimensions (number of states, number of actions) that holds the Q-values. Line 5 defines the *epsilon_greedy_action_selection* function, which implements the epsilon-greedy action selection strategy. The function returns an action that inputs the *Q_values* entity, the current state, and the epsilon value. Lines 8 to 10 define the *execute_action* function that executes the selected actions in the environment and returns the next state and reward. Lines 12 to 14 define the *update_q_value* function that updates the Q-value for the current state-action pair using the Q-learning update rule. Line 16 initializes the Q-values using the *initialize_q_values* function. Lines 18 to 27 are the main loop of the algorithm, which consists of iterating over episodes and time steps. Line 21 selects the action using the epsilon-greedy strategy, line 23 performs the selected action in the environment and obtains the next state and reward, line 25

Algorithm 1. Q-Learning Algorithm.

```
Import numpy as np
function initialize_q_values(num_states, num_actions)
    return np.zeros((num_states, num_actions))
end function
function epsilon_greedy_action_selection(Q_values, state, epsilon)
    % Implementation of epsilon-greedy action selection strategy
    % Return selected action
    pass
end function
function execute_action(action)
    % Execute the selected action in the environment
    % Return next state and reward
    pass
end function
function update_q_value(Q_values, state, action, next_state, reward, learning_rate, discount_factor)
    % Update Q-value for the current state-action pair using the Q-learning update rule
    pass
end function
Initialize Q-values
Q_values = initialize_q_values(num_states, num_actions)
% Loop over episodes
for episode in range(num_episodes) do
    % Display current system status
    display_system_status()
    % Loop over time steps
    for time_step in range(max_time_steps) do
        % Decide on an action using an exploration-exploitation strategy
        action = epsilon_greedy_action_selection(Q_values, current_state, epsilon)
        % Execute selected action in the environment
        next_state, reward = execute_action(action)
        % Update Q-value for the current state-action pair using Q-learning update rule
        update_q_value(Q_values, current_state, action, next_state, reward, learning_rate, discount_factor)
        % Move to the next state
        current_state = next_state
    end for
end for
```

updates the Q-value based on the Q-learning update formula, and line 27 shows the transition to the next state as the current state.

These steps are generally considered a reinforcement learning process in which an agent learns how to act in a specific environment. This process includes initialization, exploration and exploitation, action execution, updating the Q value, and transitioning to the next state. These steps continue as an iterative cycle until the agent converges to the desired learning or a certain number of training iterations are performed. During the training process of a QL reinforcement learning agent, the agent interacts with the environment by performing actions based on the current state and receives feedback in the form of rewards or penalties. The agent aims to learn the optimal strategy to maximize the cumulative reward over time. At each time step, the agent observes the current state of the system, selects an action based on its policy (which can be heuristic or greedy), and executes the action. After acting, the agent receives a reward based on the outcome obtained from the environment. The reward is then used to update the Q-value associated with the state-action pair using the Q-learning update rule equation:

$$Q(s,a) = Q(s,a) + \alpha \left[R + \gamma \max(Q(s',a')) - Q(s,a)\right] \tag{8}$$

$Q(s,a)$: The current estimated value of taking action a in state s. α: The learning rate (a value between 0 and 1), which determines the step size of the update. R: The immediate reward received after taking action a in state s. γ: The discount factor (between 0 and 1), which weights the importance of future rewards. $\max(Q(s',a'))$: The maximum estimated value of the next state s' over all possible actions a'. $[R + \gamma \cdot \max(Q(s',a')) - Q(s,a)]$: The temporal difference (TD) error, which measures the difference between the current estimate and the new estimate based on the reward and the best future action.

This update rule adjusts the Q-value based on the difference between the predicted reward and the actual reward plus the discounted value of the next best action, allowing the agent to learn an optimal policy over time.

After the QL reinforcement learning agent is trained on all nodes in the fog layer, it is ready to make decisions and optimize the defined objective functions in real time. The agent uses its learned knowledge and experiences to assess the current state of the system, consider available resources, and make decisions about task scheduling and resource allocation. The differential learning algorithm optimizes the objective functions by continuously evaluating the current state, taking actions based on its learned policy, re-

ceiving feedback in the form of rewards or penalties, and updating its knowledge for errors. The Q-learning algorithm updates the Q values based on the rewards received and uses these updated values to make better decisions in the future. The training and optimization process is performed on each node of the fog layer. Each node acts independently to make decisions based on its local observations and interactions with the environment. There is no need for a central server to coordinate the decision-making process, as each node is able to learn and optimize its actions independently. In general, the Q-Learning agent optimizes the defined objective functions by continuously learning and adapting to changing task requirements and system conditions. Its ultimate goal is to maximize resource utilization, minimize waste, reduce response time, and maintain the desired quality of service in the vehicular fog system.

4 Evaluation Results

This section presents the evaluation results of the proposed solution.

4.1 Dataset

Real vehicle routes are used as traffic input for the simulation. These routes are obtained from the Didi GAIA [1] open dataset, specifically from a 3×3 km^2 area in Qingyang District, Chengdu, China, on November 16, 2016 [24]. According to the dataset, four different service scenarios with different time periods are considered.

Various statistics related to the traffic data are summarized in Table 3. These statistics include the total number of vehicle trackings, the average dwell time (ADT) in the 3×3 km^2 area, the variance of the dwell time (VDT), the average number of vehicles (ANV) per second, the variance of the number of vehicles (VNV), the average speed of vehicles (ASV), and the variance of the speed of vehicles (VSV).

Certainly, for short-run tasks, the time is 193. This is because the analysis was done over short intervals and specific time periods. In this case, 5-minute intervals were chosen to capture specific traffic conditions or patterns at those times.

Scenario 1, which occurred from 8:00 to 8:05, involved a total of 718 vehicles tracked in a 3×3 km^2 area in Qingyang District. The average dwell time of these vehicles was 198.3 seconds with a variance of 123.8. The average number of vehicles passing through the area per second was 474.6 with a variance of 11.6. The average speed of vehicles in this scenario was 5.22 m/s with a variance of 2.61. This scenario represents the morning rush hour, when traffic congestion is usually high as commuters head to work or school.

Scenario 2 occurred from 13:00 to 13:05, and a total of 862 vehicles were tracked. The average dwell time was 188.5 seconds with a variance of 125.1. The average number of vehicles passing through the area per second was 541.6 with a variance of 5.38. The average vehicle speed in this scenario was 5.59 m/s with a variance of 2.73. This scenario likely represents midday traffic, where there may be a mix of commuters, delivery vehicles, and other traffic on the road. Overall, these statistics provide valuable insights into the different traffic patterns and characteristics throughout the day, which is crucial for planning and predicting the resources required for the vehicle fog system.

Scenario 3, which occurred from 18:00 to 18:05, tracked a total of 928 vehicles in a 3×3 km^2 area in Qingyang District, Chengdu, China. The average vehicle dwell time in this scenario was 196.5 seconds with a variance of 122.5 seconds. The average number of vehicles passing through the area per second was 608.0 with a variance of 7.76. The average vehicle speed was 4.60 m/s, with a variance of 2.40 m/s. This scenario represents the evening rush hour traffic in this area, with a high volume of vehicles passing through at relatively lower speeds compared to the other scenarios.

Scenario 4 was tracked from 23:00 to 23:05 with a total of 359 vehicles in a 3×3 km^2 area. The average dwell time of vehicles in this scenario was 173.7 seconds with a variance of 124.1 seconds. The average number of vehicles passing through the area per second was 207.9, with a variance of 3.93. The average vehicle speed was 7.30 m/s, with a variance of 3.16 m/s. This scenario represents late-night traffic, with fewer vehicles on the road compared to other times of the day, but with a higher average speed. The lower variance in dwell time and number of vehicles indicates a steadier flow of traffic during this time period.

4.2 Evaluation Metrics

To evaluate the performance, the following information is collected: the upload time and processing time of each task; the total number of tasks, denoted by K_{total}, and the number of tasks served, denoted by $K_{serviced}$. The total number of tasks executed locally, denoted by K_{local}. Accordingly, four metrics, namely average processing time (APT), average service time (AST), average service ratio (ASR), and cumulative reward (CR), are defined based on the following relationships.

Average processing time is a metric that calculates the average time taken to process each job. It is calculated by summing the processing time of each job and dividing it by the total number of jobs serviced.

$$APT = \frac{\sum_{i=1}^{n} Task\ processing\ time}{K_{serviced}} \tag{9}$$

[1] Https://github.com/Whale2021/Dataset

Table 3. Traffic Data Characteristics.

Traffic data								
Scenario	VSV	ASV	VNV	ANV	VDT	ADT	Trace	Time
NO.1	2.61	5.22 (m/s)	11.6	474.6	123.8	198.3 (s)	718	8:00 - 8:05 (h)
NO.2	2.73	5.59 (m/s)	5.38	541.6	125.1	188.5 (s)	862	13:00 - 13:05 (h)
NO.3	2.40	4.60 (m/s)	7.76	608.0	122.5	196.5 (s)	928	18:00 - 18:05 (h)
NO.4	3.16	7.30 (m/s)	3.93	207.9	124.1	173.7 (s)	359	23:00 - 23:05 (h)

Average service time is a metric that calculates the average time taken to complete each task. It includes both the processing time and the upload time of each task. It is calculated by summing the upload time and processing time of each task and dividing it by the total number of tasks serviced.

$$AST = \frac{\sum_{i=1}^{n} (Task\ processing\ time + upload\ time)}{K_{serviced}} \quad (10)$$

The average service ratio is a measurement that calculates the ratio of the number of tasks serviced to the total number of tasks. This indicates the effectiveness of the system in performing tasks.

$$ASR = \frac{k_{serviced}}{k_{total}} \quad (11)$$

Cumulative reward is a measure that evaluates the overall performance of the system. It is obtained by combining rewards from different criteria, such as minimizing resource waste, maximizing resource utilization, response time, and optimal quality of service.

$$CR = (\min(Resource_{wastage}) + \max(Resource_{utilization}) + \min(Response\text{-}time) + Quality\ of\ Service) \quad (12)$$

In order for different types of values and units to be summed together, they must first be normalized to a common scale. This can be achieved by assigning a weight to each criterion based on its importance, and then converting the values for each criterion to a standard score between 0 and 1. Once all the criteria have been normalized, they can be summed together to obtain an overall CR score.

Average achieved potential (AAP): It is defined as the sum of edge rewards (potential gained) over the number of edge nodes during the scheduling period, which is calculated by Eq. 13.

$$\frac{1}{E}\sum_{i=0}^{n} e\ \forall e \in E + \sum_{i=0}^{n} t\ \forall t \in T_e^{r^t} \quad (13)$$

In Eq. 13, E represents the set of all edges in a network. An edge is a connection or link between two nodes in a network. e is a distinct edge of the set E. The Equation is repeated on each edge $e \in E$. r represents the edge reward. Each edge has a reward associated with it, which can be a numerical value that represents the potential or value of that edge. t is a specific time period at the edge node. The equation is repeated at each time period $t \in T_e$, whereT_e denotes the set of all time periods associated with edge e. T represents the set of all time periods in which the computation is performed. Each edge has a set of time periods T_e associated with it.

Edges are the edges or connections between nodes in a network. In other words, edges represent possible connections between different components of a network. If the reward of an edge is high, it may indicate that the connection between two nodes in the network is very valuable and that this connection is used effectively for the intended purpose. On the other hand, if the reward of an edge is low, it may indicate that the connection between two nodes in the network is of little value. In proximal policy optimization computation, the reward value of an edge influences the decisions based on which connections between nodes are considered.

Edge reward refers to the additional reward or incentive given to an edge in a network. Edge reward is a numerical value that represents the potential or additional value of that edge. Increasing the edge reward provides the advantage of attracting more traffic or load to that particular edge and encourages the use and prioritization of that edge for communications and data transmission. This can help optimize resource utilization and improve overall system performance by reducing congestion and minimizing resource waste. On the other hand, decreasing the edge reward leads to less traffic or load being directed to that edge. This can be beneficial if other edges with better capabilities or resources should be prioritized. By decreasing the edge reward, the system can distribute the load more evenly or direct it to more optimal paths or edges.

4.3 Simulation Environment

Table 4 depicts the simulation environment. The proposed method is implemented using Python and MATLAB 2020a. The model is run on an Ubuntu 20.04 server with a 16-core AMD Ryzen 9 5950X processor clocked at 4.3 GHz, two NVIDIA GeForce RTX 3090 GPUs, and 64 GB of memory. The proposed model is run on all nodes of the fog layer. The main training parameters are configured as follows: learning rate

(α)=0.1, discount factor (λ)=0.9, and ϵ-greedy exploration rate decaying from 0.1 to 0.01. The model is trained over 100 episodes. Fog nodes are deployed in quantities ranging from 5 to 10, featuring diverse CPU and memory configurations. This heterogeneity mirrors practical vehicular fog computing scenarios. All fog nodes start with low initial utilization (20% CPU, 15% memory) to simulate a baseline empty state, with queues initialized to zero.

Table 4. System Parameter Specifications.

Parameter	**Value**
Task size	[5, 10] MB
Task compute requirement	100-500 MIPS
Computing cycles for processing 1-bit work data	500 cycles/bit
Deadline for completing tasks	[5,10]sec
V2I communication bandwidth	20MHZ
Edge node computation capability	[3,10] GHz
Maximum power consumption of V2I communications	10^3mW
V2I communication range	500m
Wired transfer rate	50Mbs
bAdditive white Gaussian noise	-90dBm
Noise power	-114dBm
User equipment bandwidth	20 MHZ
Path loss power in large scale	3dB
Simulation area	$3 \times 3\ km^2$
Learning rate (α)	0.1
Discount factor (λ)	0.9
Episodes	100
$\epsilon - greedy$	0.1 (decays to 0.01 over episodes)
Fog nodes	5–10 nodes
Fog node cpu utilization	20%
Fog node memory utilization	15%

Gigahertz refers to the frequency at which an edge node is capable of performing computations. This does not necessarily mean that the processor clock speed is 10 GHz. Computing power is not directly calculated from frequency, as the number of cores and other factors also play a role in determining the overall processing power of a system. Two processors with the same frequency but different numbers of cores can have different computing power. 1-bit work refers to processing a data unit of size 1 bit [24]. The processing power requirement of the tasks is between 100-500 MIPS. The vehicle-to-infrastructure communication bandwidth, 20 MHz, refers to the data transmission frequency, not the actual size of the data being transmitted. To calculate the data transfer rate, Shannon's law can be used, which states that the maximum data transfer rate (in bits per second) in a channel with bandwidth B and signal-to-noise ratio S/N is given by:

$$R = B * log_2(1 + \frac{S}{N}) \tag{14}$$

The simulation is performed in a $3 \times 3\ km^2$ area, with 9 edge nodes consisting of 5G base stations and roadside processing units uniformly distributed in the area, which is in accordance with [24–27]. According to [24, 25, 27, 28], the $3 \times 3\ km^2$ area provides an acceptable scale for simulating and evaluating vehicle resource allocation in a fog computing environment. This allows a sufficient number of edge nodes, in this case 9 edge nodes consisting of 5G base stations and roadside processing units, to be uniformly distributed to experiment with different resource allocation strategies. The 3×3 area provides a balance between realism and computational complexity. The computational capacities of the edge nodes, represented by their CPU clock frequencies, are varied and uniformly distributed in the range of 3–10 GHz according to [26]. The communication range for vehicle-to-infrastructure communications is set at 500 meters [26].

The proposed work is compared to Round Robin (RR) [29], First Come First Served (FCFS)[29], Weighted Fair Queuing (WFQ), and work based on Lagrange [24]. The round robin algorithm is a basic resource allocation technique that ensures fair resource allocation among different vehicles in a vehicle fog computing environment. This algorithm allocates available resources to vehicles in a round robin manner, where each vehicle is given a fixed amount of resources sequentially. The allocation continues in a round robin manner until all vehicles receive their required resources. This algorithm works well when vehicles have similar resource requirements. The advantage of the RR algorithm is that each task is executed in turn, and there is no need to wait for the previous task to complete. However, if the load is found to be heavy, the round robin takes a long time to complete all tasks. First come first serve algorithm is another traditional resource allocation method that can be applied to vehicle fog computing. In this algorithm, vehicles are serviced in the order in which they request resources. The first vehicle to request resources is allocated the available resources first, and subsequent vehicles allocate resources in the same order. The Weighted Fair Queuing scheduling algorithm is used in network and resource management to ensure fair allocation of resources among fog nodes. A fog computing environment consists of multiple fog

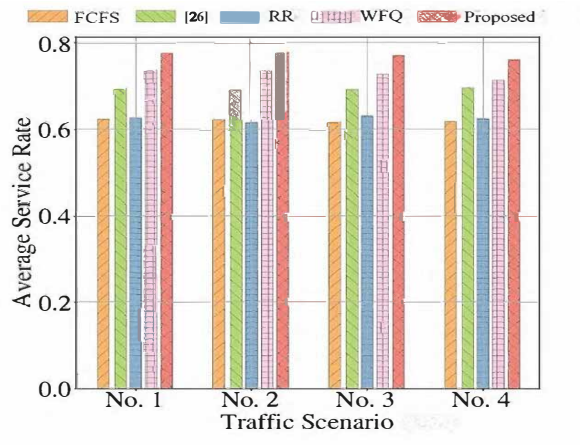
nodes that act as intermediate points between end devices and cloud servers. The aim of the paper [24] is to reduce the service time by allocating the available bandwidth. An application model is built considering the service methods and solved through a two-stage approach. For the first stage, all suboptimal solutions are presented based on the Lagrange algorithm. For the second stage, an optimal solution selection process is presented and analyzed.

4.4 Effect of Different Traffic Scenarios

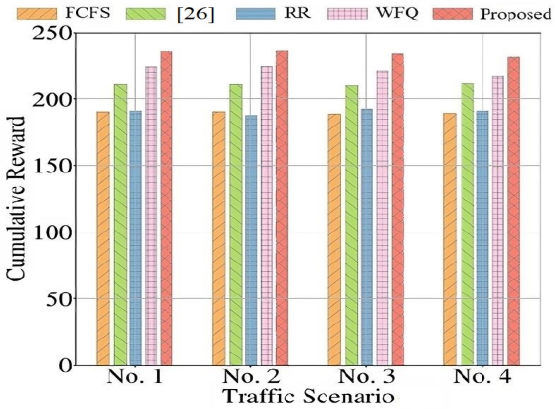
Figure 2 compares the five algorithms under different traffic scenarios. In Figure 2(a), the ASR above shows that the proposed method successfully allocates resources to vehicles and ensures that they receive the resources they need. As depicted in Figure 2(a), the average ASRs for the proposed approach are 78%, while those for FCFS, [26], RR, and WFQ are 62%, 68%, 63%, and 72%, respectively. In Figure 2(a), the Average Service Ratio (ASR) of the proposed Q-Learning method consistently outperforms traditional baselines such as FCFS, RR, WFQ, and the Lagrange-based approach [26] across all traffic scenarios, demonstrating superior task servicing even under varying densities and mobility patterns. Furthermore, the proposed method maintains stable ASR performance throughout the scenarios, highlighting its robustness in handling non-stationary traffic flows where baselines exhibit noticeable degradation due to their lack of predictive capabilities. This can be attributed to the efficient resource allocation strategy used in the proposed method that considers the needs and priorities of vehicles.

Figure 2 (b) depicts the cumulative reward for different approaches. The average CRs for the proposed approach are 235 for all scenarios, and while those are 190, 212, 191, and 225 for FCFC, [26], RR, and WFQ, respectively. The proposed Q-Learning approach consistently demonstrates the highest reward levels in all scenarios, reflecting its superior ability to balance resource demands and adapt to varying traffic conditions, while traditional baselines like FCFS, RR, WFQ, and the Lagrange-based method yield progressively lower rewards, particularly in denser or more dynamic settings. This visual trend underscores the proposed method's robustness in maximizing long-term system efficiency, as it effectively learns from interactions to prioritize high-value actions, avoiding the inefficiencies seen in static or greedy strategies that struggle with real-time variability.

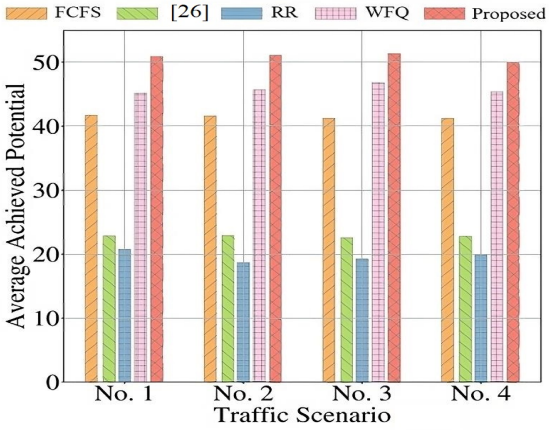
In Figure 2 (c), the proposed method is expected to achieve the highest AAP under all scenarios. Figure 2 (c) depicts the average achieved poetical (AAP) for different approaches. The average AAPs for the proposed approach are 51 for all scenarios, while those are 42, 23, 20, and 46 for FCFC, [26], RR, and WFQ,



(a) Average Service Rate.



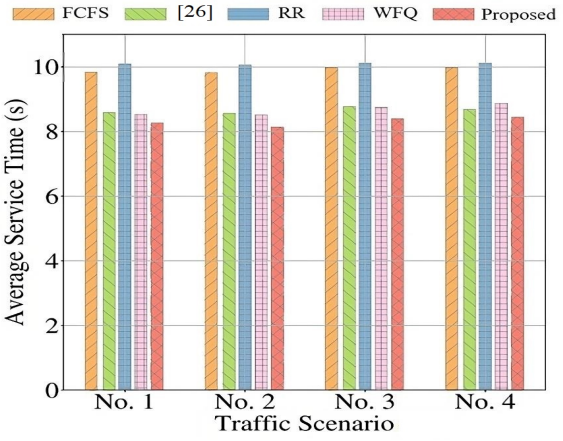
(b) Cumulative Reward.



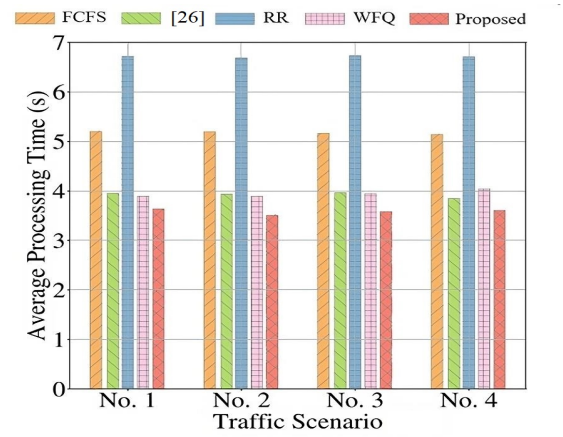
(c) Average Achieved Potential.

Figure 2. Performance Comparison for Different Scenarios.

respectively. The proposed Q-Learning approach leads with the highest value of AAP in every scenario, showing its consistent excellence in realizing system potential under diverse conditions, while baselines such as FCFS, RR, WFQ, and the Lagrange-based method [26] show lower AAP, particularly struggling in denser or more variable traffic flows. This pattern highlights the proposed method's strength in adaptive learning,



(a) Average Service Time.



(b) Average Processing Time.

Figure 3. Performance Comparison for Different Scenarios.

which maximizes resource efficiency and task fulfillment by responding fluidly to real-time demands, in contrast to the more rigid strategies of traditional approaches that falter amid changing dynamics. The higher AAP indicates that the proposed method allocates a higher proportion of available resources to vehicles and meets their resource needs more effectively. The weighting mechanism used in the method allows it to allocate resources more effectively based on the vehicle priorities.

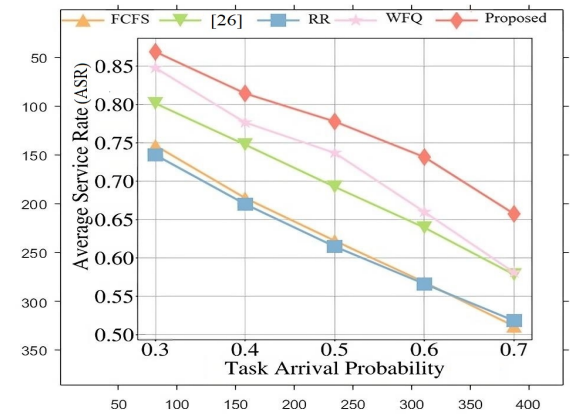
Figures 3 (a), and 3 (b) depict the average service time (AST) and average processing time (APT) across four traffic scenarios for various resource allocation methods. The proposed Q-Learning approach shows the shortest bars in both metrics throughout the scenarios, indicating its consistent efficiency in delivering quick responses and computations under different conditions, while baselines like FCFS, RR, WFQ, and the Lagrange-based method display longer bars that grow taller in denser or more variable traffic, revealing their struggles with delays. This pattern highlights the proposed method's strength in adaptive learning, which smoothly handles real-time demands to minimize wait times, in contrast to the more rigid strategies of traditional approaches that often extend processing under pressure. The superior performance of the proposed method can be attributed to the use of QL reinforcement learning, which optimizes resource allocation based on vehicle requirements and priorities. This approach enables better resource utilization, faster task processing, and efficient coverage of vehicle resource needs. In contrast, other algorithms do not perform well due to their limitations. The FCFS algorithm lacks consideration of resource needs and priorities, which leads to potentially inefficient resource allocation. The rotating turn algorithm does not dynamically adapt to the different resource needs of vehicles and can lead to unequal resource utilization. The weighted fair queuing algorithm used in [26] does not sufficiently consider vehicle priorities, which potentially leads to suboptimal resource allocation. Furthermore, [26] uses the Lagrangian algorithm, which does not obtain optimal solutions. Overall, the proposed method with reinforcement learning allows it to consider resource requirements, priorities, and achieve efficient resource allocation, thus outperforming other algorithms. This shows that, as expected, the proposed method can achieve shared communication and computation between edge nodes and improve the overall service ratio by minimizing the average service time of tasks.

4.5 The Effect of Task Arrival Probability

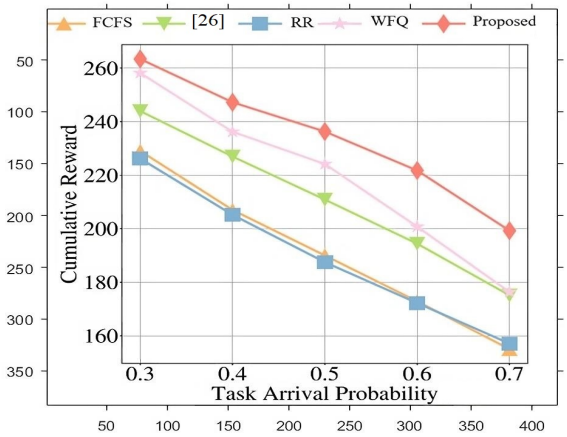
The effect of task arrival probability refers to how the frequency of sending tasks to the fog system affects its performance. In the proposed method, the performance of five algorithms under different vehicle arrival probabilities is analyzed and compared. This analysis helps to understand how different levels of task arrival rates affect resource scheduling and system forecast accuracy. By studying the effect of task arrival probabilities, the optimal strategy for efficient resource allocation in the fog system can be determined.

Figures 4, and 5 compare the five algorithms under different vehicle arrival probabilities. In this set of experiments, we consider that the vehicle arrival probability increases from 3% to 5% in each time interval. It is expected that the performance of all five algorithms deteriorates as the vehicle arrival probability increases.

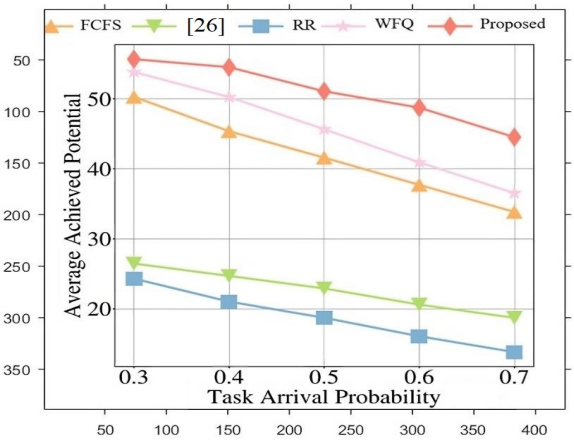
Figure 4(a) compares the ASR of the five algorithms, and the proposed method achieves the highest ASR. As the probability of vehicle arrival increases, the service rate decreases because with more vehicles arriving, the system becomes busier and servers handle more tasks simultaneously. This can lead to congestion, delay, and increased waiting time for processing each task. As a result, the average service time increases, which leads to a decrease in the service rate. In addition, servers may become overwhelmed and cannot process tasks



(a) Average Service Rate.



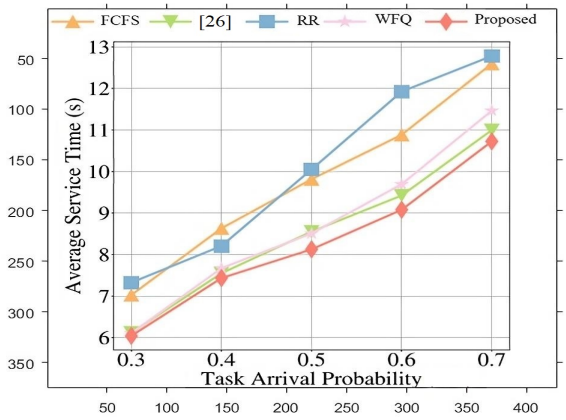
(b) Cumulative Reward.



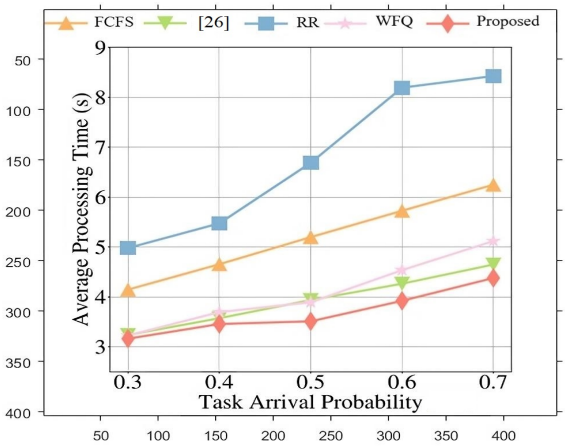
(c) Average Achieved Potential.

Figure 4. Performance Comparison Nnder Different Task Entry Probabilities.

as efficiently, which further affects the service rate. Figures 4(b) and 4(c) also compare the CR and AAP for the five algorithms and show that the proposed method can have the highest CR and AAP, which indicates the advantages of the proposed method by adopting the third-season functions as the edge node reward.



(a) Average Service Time.



(b) Average Processing Time.

Figure 5. Performance Comparison Under Different Task Entry Probabilities.

Figure 4 (b) delves into cumulative reward (CR), revealing the proposed method's marked dominance with rewards peaking at approximately 50 at low average arrival probabilities ($AAP \approx 0.3$), gradually tapering to around 20 at AAP = 0.7, yet maintaining a steeper upward curve compared to baselines like FCS (which plateaus earlier) and WFQ (showing sharper drops), underscoring the proposed algorithm's robustness in maximizing long-term value accumulation even under intensified workloads. Similarly, Figure 4(c) examines average achieved potential (AAP), where the proposed method secures the highest values—reaching nearly 35 at AAP = 0.3 and sustaining above 25 at higher loads, while competitors such as RR and [26] falter with more pronounced declines, emphasizing the innovative integration of adaptive scheduling and resource allocation in the proposed framework, which not only mitigates overload but also enhances overall system potential realization by optimizing task prioritization and server utilization.

Figures 5 (a), and 5 (b) compare the five algorithms (AST and APT). It can be seen that the performance

gap between WFQ and the proposed method in this paper [26] is small when the probability of the job arriving increases from 0.3 to 0.4. This is because the scheduling effect is not significant when there are sufficient resources. In Figures 5 (a) and 5(b), the comparison across the five algorithms reveals distinct trends in average service time and average processing time under varying task entry probabilities. The proposed method generally demonstrates superior performance, maintaining lower times compared to baselines like FCS, [26], RR, and WFQ, particularly as arrival rates intensify, where more advanced scheduling shines by efficiently allocating resources to minimize delays. Notably, the gap between WFQ and the proposed approach narrows at lower to moderate arrival probabilities, as abundant system resources diminish the impact of sophisticated scheduling, allowing simpler methods to perform comparably without significant congestion or resource contention.

The proposed method outperforms other algorithms for several reasons:

(1) Efficiency in resource allocation: The proposed method optimizes resource allocation based on a reward system considering the probability of task arrival. By assigning priority to vehicles with higher rewards, this method ensures efficient resource utilization and reduces waiting time and task delays.

(2) Adaptation to changing conditions: As the probability of tasks arriving increases, the performance gap between the proposed method and the weighted fair queue remains small. This indicates that the proposed method effectively adapts to the changing workload by dynamically adjusting the resource allocation based on the reward system.

(3) 3. Increased fairness: The weighting mechanism in the weighted fair queue may not consider the priority or importance of tasks. In contrast, the proposed method assigns higher rewards to vehicles with higher resource requirements or critical tasks. This ensures fair resource allocation and prevents low-priority tasks from affecting the performance of other vehicles.

(4) Comprehensive evaluation criteria: The analysis considers ASR, CR, AAP, AST, and APT to evaluate the performance of the algorithms. The proposed method consistently achieves higher values for these criteria, demonstrating its effectiveness in resource allocation and task scheduling.

While the proposed Q-learning-based method demonstrates superior performance in dynamic resource allocation for three-tier vehicular fog computing, it is limited by its dependency on high-quality input data for accurate predictions, the inherent complexity of the Q-learning algorithm which may lead to longer convergence times in highly volatile environments, and constraints imposed by limited hardware resources in vehicles, potentially exacerbating issues like data uncertainty and adaptation delays under extreme network congestion. Additionally, the approach assumes a relatively stable three-tier architecture, which may not fully account for real-world variabilities such as intermittent connectivity or heterogeneous vehicle capabilities, thereby restricting its scalability in ultra-large-scale deployments.

5 Conclusions

In this paper, a proposed method using QL reinforcement learning for managing and predicting the required resources of a vehicular fog system with a three-layer architecture was proposed in order to optimize resource allocation and improve the overall performance of the system. This study shows that the use of QL reinforcement learning is very effective in optimizing resource allocation, because the system can learn from past experiences, adapt to changing conditions, and make decisions based on real-time data.

One of the advantages of this method is its ability to handle dynamic and unpredictable situations, which, by using the QL reinforcement learning agent, can dynamically adjust resource allocation strategies based on the changing needs of the system. This adaptability increases the efficiency of the fog system, and by optimizing resource allocation, this scheme can potentially increase the overall reliability and availability of the automotive fog system, leading to improved user experience and satisfaction. The findings of this research can have practical implications for the development of more effective automotive fog computing architectures and resource management strategies that benefit both industry and academia.

Despite its high reliability and favorable performance in real-world environments, this method faces challenges, limitations, and weaknesses that require attention and improvement. These challenges include: Dependency on input data, hardware resource limitations in vehicles, complexity of the Q-learning algorithm, data uncertainty, and adapting to changing conditions. For future work, integrating hybrid reinforcement learning techniques, such as combining Q-learning with deep neural networks, could enhance convergence speed and robustness; conducting real-world pilot implementations in diverse urban scenarios would validate and refine the model; and exploring multi-objective optimization to balance energy efficiency alongside performance metrics would broaden its applicability in sustainable IoV ecosystems.

References

[1] M. Nadeem Ahangar, Qasim Z. Ahmed, Fahd A. Khan, and Maryam Hafeez. A survey of autonomous vehicles: Enabling communication technologies and challenges. *Sensors*, 21(3), 2021. ISSN 1424-8220. doi: 10.3390/s21030706. URL https://www.mdpi.com/1424-8220/21/3/706.

[2] Sekione Reward Jeremiah, Laurence Tianruo Yang, and Jong Hyuk Park. Digital twin-assisted resource allocation framework based on edge collaboration for vehicular edge computing. *Future Generation Computer Systems*, 150: 243–254, 2024. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2023.09.001. URL https://www.sciencedirect.com/science/article/pii/S0167739X23003278.

[3] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology*, 68(4):3113–3125, 2019. doi: 10.1109/TVT.2019.2894851.

[4] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. Computation task assignment in vehicular fog computing: A learning approach via neighbor advice. In *2019 IEEE 18th International Symposium on Network Computing and Applications (NCA)*, pages 1–5, 2019. doi: 10.1109/NCA.2019.8935033.

[5] Chao Zhu, Jin Tao, Giancarlo Pastor, Yu Xiao, Yusheng Ji, Quan Zhou, Yong Li, and Antti Ylä-Jääski. Folo: Latency and quality optimized task allocation in vehicular fog computing. *IEEE Internet of Things Journal*, 6(3):4150–4161, 2019. doi: 10.1109/JIOT.2018.2875520.

[6] Jinming Shi, Jun Du, Jian Wang, and Jian Yuan. Deep reinforcement learning-based v2v partial computation offloading in vehicular fog computing. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2021. doi: 10.1109/WCNC49053.2021.9417450.

[7] Vivek Sethi and Sujata Pal. Feddove: A federated deep q-learning-based offloading for vehicular fog computing. *Future Generation Computer Systems*, 141:96–105, 2023. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2022.11.012. URL https://www.sciencedirect.com/science/article/pii/S0167739X22003739.

[8] Mashael Khayyat, Ibrahim A. Elgendy, Ammar Muthanna, Abdullah S. Alshahrani, Soltan Alharbi, and Andrey Koucheryavy. Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks. *IEEE Access*, 8:137052–137062, 2020. doi: 10.1109/ACCESS.2020.3011705.

[9] Seung-Seob Lee and Sukyoung Lee. Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information. *IEEE Internet of Things Journal*, 7 (10):10450–10464, 2020. doi: 10.1109/JIOT.2020.2996213.

[10] Junhui Zhao, Ming Kong, Qiuping Li, and Xiaoke Sun. Contract-based computing resource management via deep reinforcement learning in vehicular fog computing. *IEEE Access*, 8:3319–3329, 2020. doi: 10.1109/ACCESS.2019.2963051.

[11] Xianjing Wu, Shengjie Zhao, and Hao Deng. Joint task assignment and resource allocation in vfc based on mobility prediction information. *Computer Communications*, 205:24–34, 2023. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2023.04.004. URL https://www.sciencedirect.com/science/article/pii/S0140366423001172.

[12] Habtamu Mohammed Birhanie, Mohammed Ayoub Messous, Sidi-Mohammed Senouci, El-Hassane Aglzim, and Ahmedin Mohammed Ahmed. Mdp-based resource allocation scheme towards a vehicular fog computing with energy constraints. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, 2018.

[13] Yifan Zhang, Xiaoqi Qin, and Xianxin Song. Mobility-aware cooperative task offloading and resource allocation in vehicular edge computing. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6, 2020. doi: 10.1109/WCNCW48565.2020.9124825.

[14] Lei Liu, Jie Feng, Xuanyu Mu, Qingqi Pei, Dapeng Lan, and Ming Xiao. Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems*, 24(12):15513–15526, 2023. doi: 10.1109/TITS.2023.3249745.

[15] Seung-Seob Lee and Sukyoung Lee. Poster abstract: Deep reinforcement learning-based resource allocation in vehicular fog computing. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1029–1030, 2019. doi: 10.1109/INFCOMW.2019.8845095.

[16] Xianjing Wu, Shengjie Zhao, Rongqing Zhang, and Liuqing Yang. Mobility prediction-based joint task assignment and resource allocation in vehicular fog computing. In *2020 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2020. doi: 10.1109/WCNC45663.2020.9120524.

[17] Liangliang Yan, Min Zhang, Chuang Song, Danshi Wang, Jin Li, and Luyao Guan. Deep learning-based containerization resource management in vehicular fog computing. In *2019 Asia Communications and Photonics Conference (ACP)*, pages 1–3, 2019.

[18] Ashok Sutagundar, Ameenabegum H Attar, and Basamma Patil. Resource allocation for fog enhanced vehicular services (fevs). In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 360–365, 2018. doi: 10.1109/ICIRCA.2018.8597428.

[19] Lin Gu, Deze Zeng, Song Guo, Ahmed Barnawi, and Yong Xiang. Cost efficient resource management in fog computing supported medical cyber-physical system. *IEEE Transactions on Emerging Topics in Computing*, 5(1):108–119, 2017. doi: 10.1109/TETC.2015.2508382.

[20] Tingting Liu, Jun Li, Feng Shu, and Zhu Han. Optimal task allocation in vehicular fog networks requiring urllc: An energy-aware perspective. *IEEE Transactions on Network Science and Engineering*, 7(3):1879–1890, 2020. doi: 10.1109/TNSE.2019.2955474.

[21] Fuhong Lin, Yutong Zhou, Giovanni Pau, and Mario Collotta. Optimization-oriented resource allocation management for vehicular fog computing. *IEEE Access*, 6:69294–69303, 2018. doi: 10.1109/ACCESS.2018.2879988.

[22] Fatin Hamadah Rahman, S.H. Shah Newaz, Thien-Wan Au, Wida Susanty Suhaili, M.A. Parvez Mahmud, and Gyu Myoung Lee. Entruve: Energy and trust-aware virtual machine allocation in vehicle fog computing for catering applications in 5g. *Future Generation Computer Systems*, 126:196–210, 2022. ISSN 0167-739X. doi: https://doi.org/10.1016/j.future.2021.07.036. URL https://www.sciencedirect.com/science/article/pii/S0167739X21002983.

[23] Chinmaya Kumar Dehury, Bharadwaj Veeravalli, and Satish Narayana Srirama. Herafc: Heuristic resource allocation and optimization in multifog-cloud environment. *Journal of Parallel and Distributed Computing*, 183: 104760, 2024. ISSN 0743-7315. doi: https://doi.org/10.1016/j.jpdc.2023.104760. URL https://www.sciencedirect.com/science/article/pii/S0743731523001302.

[24] Chunhui Liu, Kai Liu, Hualing Ren, Xincao Xu, Ruitao Xie, and Jingjing Cao. Rtds: real-time distributed strategy for multi-period task offloading in vehicular edge computing environment. *Neural Comput. Appl.*, 35(17): 12373–12387, March 2021. ISSN 0941-0643. doi: 10.1007/s00521-021-05766-5. URL https://doi.org/10.1007/s00521-021-05766-5.

[25] Bushra Jamil, Humaira Ijaz, Mohammad Shojafar, and Kashif Munir. Irats: A drl-based intelligent priority and deadline-aware online resource allocation and task scheduling algorithm in a vehicular fog network. *Ad Hoc Networks*, 141:103090, 2023. ISSN 1570-8705. doi: https://doi.org/10.1016/j.adhoc.2023.103090. URL https://www.sciencedirect.com/science/article/pii/S1570870523000100.

[26] Zhenyu Zhou, Pengju Liu, Junhao Feng, Yan Zhang, Shahid Mumtaz, and Jonathan Rodriguez. Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach. *IEEE Transactions on Vehicular Technology*, 68(4):3113–3125, 2019. doi: 10.1109/TVT.2019.2894851.

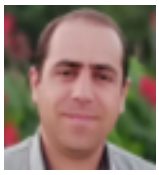
[27] Lianming Xu, Zexuan Yang, Huaqing Wu, Yanru Zhang, Yanhui Wang, Li Wang, and Zhu Han. Socially driven joint optimization of communication, caching, and computing resources in vehicular networks. *IEEE Transactions on Wireless Communications*, 21(1):461–476, 2022. doi: 10.1109/TWC.2021.3096881.

[28] Hongbiao Zhu, Qiong Wu, Xiao-Jun Wu, Qiang Fan, Pingyi Fan, and Jiangzhou Wang. Decentralized power allocation for mimo-noma vehicular edge computing based on deep reinforcement learning. *IEEE Internet of Things Journal*, 9 (14):12770–12782, 2022. doi: 10.1109/JIOT.2021.3138434.

[29] Dr. Amit Agarwal and Saloni Jain. Efficient optimal algorithm of task scheduling in cloud computing environment, 2014. URL https://arxiv.org/abs/1404.2076.

Bahar Mojtabaei received her B.S. degree in Computer Engineering from Lorestan University and subsequently received her M.Sc. degree in Computer Architecture from Razi University in 2024. Her research interests include vehicular computing, fog computing, and machine learning.

Mahmood Ahmadi received the B.S. degree in Computer Engineering from the University of Isfahan, Isfahan, Iran, in 1995. He received the M.Sc. degree in Computer Architecture and Engineering from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 1998. From 1999 to 2005, he was a faculty member at Razi University in Kermanshah. In October 2005, he joined the Faculty of Electrical Engineering, Mathematics, and Computer Science (EEMCS) at Delft University of Technology, Delft, The Netherlands, as a full-time Ph.D. student. He received his Ph.D. in May 2010. His research interests include Software-Defined Networking, Named-Data Networking, Network Function Virtualization, Fog and Cloud Computing, the Internet of Things, Intrusion Detection, and High-Performance Computing. He has published more than 110 journal and conference papers. He is currently a Professor at the Computer Engineering Department, Razi University, Kermanshah, Iran.



Sajad Ahmadian received his B.S. degree in computer engineering from Razi University, Kermanshah, Iran, in 2011, and the M.Sc. degree in computer engineering, artificial intelligence from University of Kurdistan, Sanandaj, Iran, in 2014 and a PhD Ph.D. degree in computer engineering, artificial intelligence from University of Zanjan, Zanjan, Iran, in 2018. He conducted a part of his Ph.D. research work in the laboratory of complex networks, RMIT University, Melbourne, Australia, form February 2018 to July 2018. He is currently the acting Dean at the Faculty of Information Technology, Kermanshah University of Technology, Kermanshah, Iran. His research interests include recommender systems, social networks analysis, data mining, Healthcare informatics, and machine learning where he led several research projects. In 2022, he visited University of Oulu as part of DigiHealth Visiting programme.