# Optimizing Task Offloading in Vehicular Network (OTO): A Game Theory Approach Integrating Hybrid Edge and Cloud Computing

**Mohanapriya .M[1,*], V. Anusuya[2], K. Aravindhan[3], N. Krishnaveni[4], R. Santhosh[5], D. Gowthami[6]**

[1]Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore, India

[2]Department of Information Technology, Ramco Institute of Technology, North Venganallur village, Rajapalayam , Tamil Nadu, India

[3]School of Computing and Information and Technology, Reva University, Bangalore, India

[4]Computer Science and Engineering, Vel Tech Rangarajan Dr.Sagunthala R&D Institute of Science and Technology, Chennai, Tamilnadu, India

[5]Department of Computer Science and Engineering, Faculty of Engineering, Karpagam Academy of Higher Education, Coimbatore, Tamil Nadu, India

[6]Department of Computer Science and Engineering, Karpagam College of Engineering, Coimbatore, Tamil Nadu, India

Emails: mohanapriyaasaithambi@gmail.com; pgkrishanu@gmail.com; aravindhan.kurunthachalam@reva.edu.in; santhoshrd@gmail.com; gowthamime16@gmail.com; nkrishnaveni984@gmail.com

**Abstract**

In VANETs, user equipment (UE) schedules tasks by prioritizing them based on urgency and resource availability to ensure timely and efficient communication and processing. Effective task scheduling and resource allocation in VANET are crucial for maintaining low latency, high reliability, and optimal resource utilization for real-time vehicular communications. However, existing works often face limitations such as inadequate handling of dynamic network conditions, leading to increased latency and suboptimal resource usage. In this paper, we introduced a precise model by proposing Optimizing Task Offloading in Vehicular Network named as OTO framework. Initially, UEs are clustered using an Improved Fuzzy Algorithm (IFA) to reduce latency and energy consumption, with optimal clusters determined by a cluster validity index. Clustering considers distance, location, RSSI, link stability, and trust values, and cluster heads (CH) chosen based on distance, trust, and link stability. Following this, tasks from UE are classified using a Hybrid Deep Learning (HDL) algorithm, with LiteCNN for classification into emergency and non-emergency tasks and LiteLSTM for scheduling to reduce the weight matrix and overfitting. Dual scheduling based on task length, delay sensitivity, QoS, priority, resource consumption, and queue length reduces execution time and latency. Finally, the scheduled tasks are allocated to the optimal edge server based on task load, resource availability, waiting time, and distance using the RL-based Multi-agent Deep Reinforcement Learning (MA-DRL) algorithm, where edge servers act as sellers and users as buyers, reducing latency due to high convergence. In order to, evaluate and prove the efficacy of proposed OTO framework, we performed comparative analysis in terms of several performance metrics where our proposed OTO model outperforms other existing approaches.

**Keywords:** Task Scheduling; Game Theory; Resource Allocation; Task Classification; Hybrid Deep Learning (HDL); Multi-agent Deep Reinforcement Learning (MA-DRL)

## 1. Introduction

The evolution of vehicular networks has paved the way for significant advancements in intelligent transportation systems, autonomous driving, and connected vehicles [1]. These networks enable vehicles to communicate with each other and with infrastructure elements, facilitating a range of applications such as real-time traffic management, vehicle-to-everything (V2X)



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

communications, and advanced driver-assistance systems (ADAS). However, the increasing complexity and computational demands of these applications pose substantial challenges in terms of resource management and computational efficiency. This has led to the exploration of task offloading and resource allocation strategies within the hybrid edge-cloud computing paradigm [2].Task offloading is the process of transferring computational tasks from resource-constrained vehicular devices to more powerful external computing resources, such as edge servers (fog nodes) or cloud servers. In vehicular networks, offloading can significantly enhance application performance by reducing computational latency, conserving energy, and improving the overall user experience [3].Edge computing involves deploying computing resources at the edge of the network, closer to the end-users. In the context of vehicular networks, edge nodes (fog nodes) can be strategically placed at road intersections, traffic lights, or within vehicles themselves. These nodes can perform computational tasks locally, thus reducing the latency associated with cloud computing and enabling real-time processing for latency-sensitive applications [4].Cloud computing provides virtually unlimited computational resources located at centralized data centers. Offloading tasks to the cloud allows for handling more complex and resource-intensive computations that cannot be processed efficiently at the edge. However, this approach may introduce higher latency due to the physical distance between the vehicles and the cloud servers [5].

A hybrid approach leverages the strengths of both edge and cloud computing. Tasks are dynamically offloaded to either edge nodes or cloud servers based on their computational requirements, latency sensitivity, and current network conditions. This approach aims to optimize resource utilization and improve the quality of service (QoS) for vehicular applications by balancing the trade-offs between latency and computational power [6].Resource allocation in vehicular networks involves distributing available computing, storage, and communication resources among various tasks and applications to maximize efficiency and performance. Effective resource allocation is crucial for ensuring that offloaded tasks are processed in a timely and efficient manner. Allocating CPU and memory resources to offloaded tasks based on their computational requirements and priority [7] [8]. This includes dynamically adjusting resource allocations to match the varying demands of different applications, ensuring that critical tasks receive the necessary resources for prompt execution. Managing bandwidth and network connectivity to ensure reliable and low-latency communication between vehicles, edge nodes, and cloud servers [9][10]. This involves optimizing the use of available wireless communication channels, such as LTE, 5G, and dedicated short-range communications (DSRC), to maintain seamless data transmission.

Allocating storage space for data generated by vehicular applications, such as sensor data, video streams, and telemetry information. Efficient storage management ensures that data is readily available for processing and analysis, enabling real-time decision-making and enhanced situational awareness. Despite the potential benefits of task offloading and resource allocation in vehicular networks, several limitations and challenges need to be addressed to realize their full potential [11][12].Many vehicular applications, such as collision avoidance and autonomous driving, require real-time processing with minimal latency. Ensuring low-latency communication and computation is challenging, especially in dynamic and unpredictable vehicular environments. Vehicular networks comprise diverse devices with varying computational capabilities, communication interfaces, and energy constraints [13]. Managing this heterogeneity and ensuring efficient resource utilization across different nodes is a complex task. High vehicle mobility and dynamic network conditions can lead to frequent changes in connectivity and resource availability. Developing robust offloading and resource allocation strategies that can adapt to these fluctuations is critical for maintaining consistent performance [14].Offloading tasks and data to external servers raises concerns about data security and user privacy. Ensuring secure communication channels, protecting sensitive information, and implementing robust authentication mechanisms are essential to prevent unauthorized access and data breaches [15] – [20].

Game theory-based hybrid task offloading effectively addresses the limitations of traditional offloading strategies in vehicular networks by leveraging strategic decision-making frameworks to optimize the allocation of computational tasks between edge and cloud resources. This approach minimizes latency and meets real-time requirements by dynamically selecting optimal offloading nodes based on current network conditions and task demands, using models such as Stackelberg games to prioritize low-latency options. It manages resource heterogeneity through cooperative game theory, where vehicles and computing nodes form coalitions to share resources efficiently, and leveraging diverse node capabilities. Mobility and network dynamics are tackled with repeated games and Markov decision processes (MDPs), enabling continuous adaptation to changing conditions and maintaining consistent performance despite high mobility and network volatility. Security and privacy are enhanced by non-cooperative game theory, modeling interactions with potential attackers to develop optimal defines strategies and secure communication channels, while incentive-compatible mechanisms promote honest behavior and cooperation among nodes. Scalability and resource management are improved using auction-based models and market-driven approaches, allowing nodes to bid for computational power based on needs and priorities, ensuring fair and efficient resource distribution even in large-scale deployments. Energy efficiency is optimized through evolutionary games that balance the trade-off between computational offloading and energy usage, developing strategies that prolong the operational life of vehicular devices. Finally, interoperability is enhanced by establishing standardized protocols and incentives for cooperation among different vehicular platforms, edge nodes, and cloud servers, ensuring seamless communication and data exchange. In summary, game theory-based hybrid task offloading provides robust solutions to the inherent



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

challenges of vehicular networks, promoting smarter, more efficient, and integrated transportation systems by addressing latency, resource heterogeneity, mobility, security, scalability, energy efficiency, and interoperability.

### A. Motivation & Objectives

An IoT-Edge-Cloud environment encounters several challenges in task offloading due to mobility, security threats, and inefficient target detection. While existing solutions address some of these issues, they do not provide optimal results. This research focuses on the following problems:

- High Latency - Tasks collected from individual IoT devices often experience high latency due to their heterogeneous nature. Many existing approaches select random target servers for task allocation and offloading, rather than optimal servers, resulting in increased latency.
- Low Throughput - Failure to consider priority, QoS, and resource consumption in task scheduling leads to inefficient scheduling, reducing throughput. Most existing works overlook device mobility, causing poor scheduling and offloading, increased data loss, and reduced throughput.
- High energy consumption- It is significant limitation in the IoT-Edge-Cloud environment. The need to constantly transmit data between IoT devices, edge nodes, and cloud servers drains battery life quickly. Inefficient task scheduling and offloading strategies further exacerbate this issue, leading to unnecessary energy use. Additionally, high energy consumption impacts the overall sustainability and operational costs of the IoT-Edge-Cloud system.

### B. Research Contribution

To achieve betterment task scheduling and resource allocation approach we proposed multiple novel contributions which are articulated as follows,

- **Improved Clustering with IFA and Block chain Integration:** The proposed OTO framework enhances the clustering process of UE in vehicular networks using an IFA, considering multiple metrics such as distance, location, RSSI, and link stability to optimize latency and energy consumption.
- **Hybrid Deep Learning for Task Classification and Scheduling:** The framework introduces anHDL algorithm combining LiteCNN and LiteLSTM, which classifies tasks into emergency and non-emergency categories and schedules them efficiently by reducing the weight matrix and overfitting, resulting in improved execution time and reduced latency.
- **Efficient Task Offloading with MA-DRL:** The OTO framework employs a novel RL-based MA-DRL algorithm for optimal task allocation to edge servers, modelled as a buyer-seller game, which enhances resource utilization and reduces latency through high convergence rates.

### C. Paper Organization

The following tasks are structured as outlined: Section II provides a comprehensive overview of the task scheduling and identifies its research gaps. Section III depicts the network model of proposed OTO. Section IV details the research methodology, including relevant theoretical, diagrammatical, and mathematical explanations. In Section IV, the experimental results are presented, and evaluation outcomes compared with the latest versions of task scheduling, task offloading and resource allocation models. Section V concludes the research.

## 2. Related works

Author in [21], presents a novel approach to computation offloading in heterogeneous vehicular networks using deep reinforcement learning (DRL). The authors effectively leverage DRL to dynamically adapt offloading decisions, accommodating the diverse and dynamic nature of vehicular networks. The study is well-structured, with comprehensive experiments that demonstrate significant improvements in offloading efficiency and performance. However, while the approach shows promise, the complexity of the DRL model and its training requirements may pose practical implementation challenges. Future work could explore simplifying the model and reducing the computational overhead for real-world applications. Author in [22], focuses on reducing both delay and cost in computation offloading within vehicular edge computing environments. The paper presents a clear and concise formulation of the problem, followed by an innovative solution that balances these critical metrics. The proposed method is validated through extensive simulations, showing a substantial decrease in both delay and cost compared to traditional methods. One limitation is the



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

assumption of perfect knowledge of network conditions, which may not always be feasible in real-world scenarios. Future research could address this by incorporating predictive models or adaptive mechanisms to handle uncertainty in network conditions. In this paper [23], the authors propose an advanced deep learning-based approach for computational offloading in multilevel vehicular edge-cloud computing networks. The multi-tier architecture effectively distributes computation tasks between edge and cloud resources, optimizing resource utilization and reducing latency. The use of deep learning models to predict offloading decisions enhances the system's adaptability to varying network conditions. The experimental results are compelling, showing marked improvements over existing methods. However, the reliance on extensive training data for the deep learning model could be a drawback, and further exploration into reducing data dependency would be beneficial. Author in [24], explores partial computation offloading combined with adaptive task scheduling in 5G-enabled vehicular networks. The approach allows for more flexible and efficient use of network resources, taking advantage of 5G's high bandwidth and low latency. The authors provide a detailed analysis of the system's performance, demonstrating significant enhancements in both computational efficiency and task completion times. The partial offloading strategy is particularly innovative, offering a balanced trade-off between local computation and offloading. However, the paper could benefit from a deeper discussion on the potential impact of varying vehicular mobility patterns on the proposed system's performance. Author in [25], addresses the critical issue of energy efficiency in vehicular edge cloud computing through optimized computation offloading. The authors present a comprehensive model that minimizes energy consumption while maintaining computational performance. The proposed solution incorporates both energy and latency considerations, providing a balanced approach to offloading decisions. Experimental results show significant energy savings compared to conventional methods. One area for improvement could be the exploration of real-time implementation challenges, as the current model assumes ideal conditions. Future work could investigate adaptive algorithms that respond to real-time changes in vehicular network environments.

The proposed algorithm balances energy consumption with SLA requirements, optimizing offloading decisions across both edge and cloud resources. The authors provide a thorough analysis and extensive simulations, demonstrating the algorithm's effectiveness in reducing energy consumption while meeting SLA constraints. However, the computational complexity of genetic algorithms and their convergence time might be concerns for real-time applications. Future work could focus on enhancing the algorithm's efficiency and exploring hybrid approaches that combine genetic algorithms with other optimization techniques.

## 3. Methodology

Two persistently challenging problems in dynamic multi-user virtual networks (VNs) are network congestion and high vehicle mobility. When numerous vehicles simultaneously transfer their computing tasks to the same edge servers, it leads to increased processing times due to congestion. Consequently, this diminishes vehicle performance. Therefore, assigning computing tasks to the nearest edge server is not always the optimal solution. Figure 1 illustrates such scenarios, showing that $RSU_1$ becomes overloaded due to heightened offloading demands. The total workload that $\mathcal{N}$ vehicles have offloaded to $RSU_i$ is then calculated as:

$$\psi_i^{MEC} = \sum_{i=1}^{\mathbb{m}} \mu\varpi_i$$

Thus, the remaining computational capability of the MEC server, $\sigma_i$ can be determined using Equation (2).

$$\sigma_i = \mathbb{F}_i^{MEC} - \psi_i^{MEC}$$

Due to the overloaded issue, when $\sigma_i < 0$, the MEC server will need additional resources to complete the job. In this



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

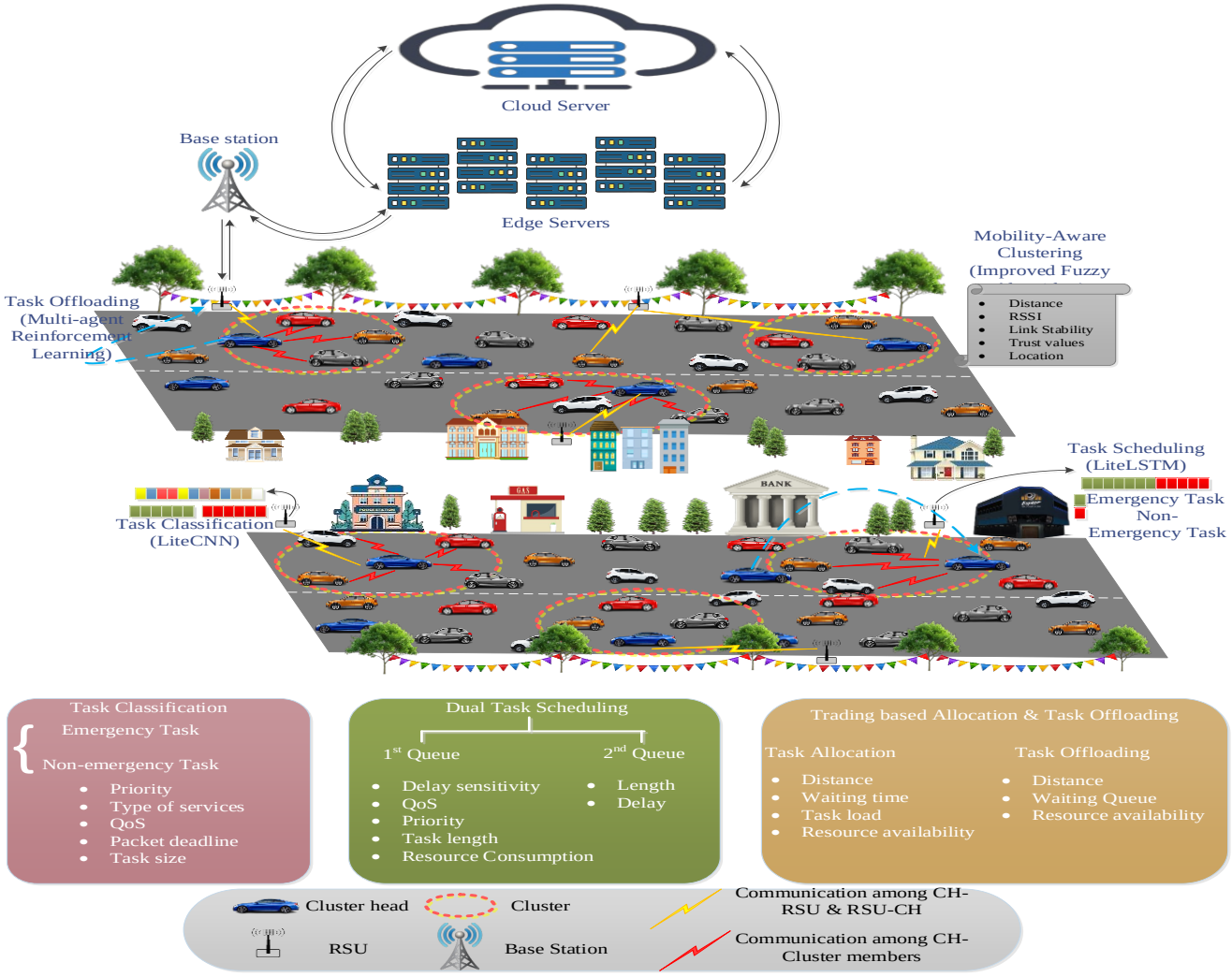


**Figure 1.** Overall proposed OTO Framework

Situation, it becomes challenging to decide whether to transfer the computational task to a distant server or keep it on the MEC server. To address the overloaded issue of $RSU_1$ and reduce reaction time, we propose an effective work offloading strategy based on game theory. In our suggested game model, each vehicle decides whether to offload computational tasks to a cloud server or a MEC server for processing. However, one of the main challenges with VNs is the high mobility of vehicles. Several scenarios depicted in Figure 1 are described below.

- State 1: Vehicles V1 and V2 enter $RSU_1$ coverage area and offload their tasks to the corresponding MEC server. While the MEC server successfully processes the task for vehicle V1, it encounters an overload issue when executing tasks for vehicle V2.
- State 2: Vehicle V3 enters $RSU_2$ coverage area and offloads a task to the corresponding MEC server. Before completing this task, the vehicle has moved through several RSU coverage areas. Once the task is completed, the vehicle enters the $RSU_M$ coverage area.

Let $\mathcal{L} = \{1,2,3,..,\mathcal{L}\}$ represent the collection of $\mathcal{L}$ vehicles in the proposed model. Assuming each vehicle generates $\mathcal{A}$ distinct tasks upon arrival, we denote the tasks as $\mathcal{A} = \{\mathcal{A}_i | i = 1,2,3,..,\mathcal{A}\}$. Each computation task $\mathcal{A}_i$ is characterized by $\mathcal{A}_i = \{\varsigma_i^{inp}, \mathbb{c}_i, \mathbb{t}_i^{max}\}$. $\varsigma_i^{inp}$ indicates the size of the input job for task $\mathcal{A}_i$. $\mathbb{c}_i$ represents the required computing resources for task processing.$\mathbb{t}_i^{max}$ denotes the maximum acceptable latency for task $\mathcal{A}_i$. Each task can be completed in one of three ways: on the cloud server via the BS, on a cloud server via an RSU, or on the MEC server.Each truck has three options for offloading tasks: to the cloud server via the BS, to a cloud server via an RSU, or to the MEC server. Our proposed architecture considers a unidirectional



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

route with $\mathcal{B}\mathrm{RSU_s}$ positioned at equal intervals, each having the same coverage area. The collection of RSUs is denoted as $\mathrm{RSU} = \{\mathrm{RSU_1}, \mathrm{RSU_2}, .., \mathrm{RSU_{\mathcal{B}}}\}$, where $\{\mathrm{RSU_i} | i = 1,2,..,\mathcal{B}\}$.The road is divided into $\mathcal{B}$ segments of length $\mathbb{L}$, denoted by $\{\mathbb{L}_\mathrm{i} | i = 1,2,..,\mathcal{B}\}$, with vehicles randomly dispersed over these segments. Each $\mathrm{RSU_{\mathbb{b}}}$ can be accessed only when vehicles are within the corresponding $\mathbb{b}$th segment. Every RSU has a communication range of 500 meters. Each RSU is equipped with a single MEC server, which has limited processing power and storage space to handle vehicle task offloading. The formula for each MEC server $\mathrm{MEC_i}$ is represented as $\mathrm{MEC_i} = \left\{\mathbb{F}_\mathrm{i}^{\mathrm{MEC}}, \mathrm{s_i^{MEC}}\right\}$, where $\mathbb{F}_\mathrm{i}^{\mathrm{MEC}}$ and $\mathrm{s_i^{MEC}}$ denote the computational and storage capacities of $\mathrm{MEC_i}$ respectively.

**Interaction and Computation Model:**

Our proposed model states that the offloaded operation can be completed by either an edge server or a remote cloud. In both cases, the overall job completion time is the sum of the task communication and processing times.

**Edge Offloading:**

For offloading calculation job $\mathcal{A}_i$, when vehicle $i$ selects a MEC server linked to $\mathrm{RSU_{\mathbb{b}}}$, the overall processing time can be calculated as follows:

$$\mathbb{t}_{i,\mathbb{b}}^{MEC} = \alpha_{i,\mathbb{b}}^{up} + \alpha_{i,\mathbb{b}}^{exe} + \alpha_{i,\mathbb{b}}^{down}$$

where vehicle $i$ offloads its computational job to $\mathrm{RSU_{\mathbb{b}}}$, which is linked to the MEC server. Here, $\alpha_{i,\mathbb{b}}^{down}$, $\alpha_{i,\mathbb{b}}^{up}$ indicates the downlink and uplink delay during transmission, and the $i$th vehicle receives the results. Additionally, $\alpha_{i,\mathbb{b}}^{exe}$ represents the time it takes for vehicle $i$ job to be processed on the MEC server. This time is calculated as follows:

$$\alpha_{i,\mathbb{b}}^{exe} = \frac{\varsigma_i^{inp}}{\mathbb{C}_{\mathbb{b}}^{MEC}}\left(1 - \mathbb{U}_{\mathbb{b}}^{MEC}\right)$$

In this case, the job size is denoted by $\varsigma_i^{inp}$, and the computational power and usage of $\mathrm{RSU_{\mathbb{b}}}$ attached to the MEC server are indicated by $\mathbb{C}_{\mathbb{b}}^{MEC}$ and $\mathbb{U}_{\mathbb{b}}^{MEC}$, respectively.

**Cloud Offloading:**

In our proposed approach, vehicles have two options to offload and compute their tasks at a remote server. When a vehicle offloads its work to a remote server via an RSU, the task's total processing time can be calculated as follows:

$$\mathbb{t}_{i,RSU}^{cloud} = \alpha_{i,cRSU}^{up} + \alpha_{i,cRSU}^{exe} + \alpha_{i,cRSU}^{down}$$

where the uplink delay via the RSU indicates $\alpha_{i,cRSU}^{up}$ and downlink delays via the RSU indicates $\alpha_{i,cRSU}^{down}$. Additionally, the processing time to complete the work on the cloud is represented by $\alpha_{i,cRSU}^{exe}$. On the other hand, if vehicle $i$ uses the BS to offload the work to a distant server, the total processing time for vehicle $i$'s task can be calculated as follows:

$$\mathbb{t}_{\mathrm{i,BS}}^{\mathrm{cloud}} = \alpha_{\mathrm{i,cBS}}^{\mathrm{up}} + \alpha_{\mathrm{i,cBS}}^{\mathrm{exe}} + \alpha_{\mathrm{i,cBS}}^{\mathrm{down}}$$

where the uplink delay via the RSU indicates $\alpha_{\mathrm{i,cBS}}^{\mathrm{up}}$ and downlink delays via the RSU indicates $\alpha_{\mathrm{i,cBS}}^{\mathrm{down}}$. Additionally, in the cloud, the task execution time is represented by $\alpha_{\mathrm{i,cBS}}^{\mathrm{exe}}$. The execution time $\alpha_{\mathrm{i,cBS}}^{\mathrm{exe}}$ is the same in both scenarios (task offloading through RSU or BS) and can be computed as follows:

$$\alpha_{i,c}^{exe} = \frac{\varsigma_i^{inp}}{\mathbb{C}^{cloud}}\left(1 - \mathbb{U}^{cloud}\right)$$

In this case, $\mathbb{U}^{cloud}$ and $\mathbb{C}^{cloud}$ represent the cloud server's average utilization and computing capacity, respectively.

**A. Mobility-aware Clustering**

Determining the optimal number $\mathbb{c}$ of clusters is crucial because as $\mathbb{c}$ increases, the amount of inter-cluster communication also increases. Conversely, with a smaller $\mathbb{c}$, there is a higher volume of intra-cluster communications. To find the optimal number of clusters, we will use the silhouette score technique [47], or the silhouette coefficient (SC), as follows:

$$\mathrm{SC}(\eta_\mathrm{i}) = \frac{\mathfrak{b}(\eta_\mathrm{i}) - \mathfrak{a}(\eta_\mathrm{i})}{max\{\mathfrak{a}(\eta_\mathrm{i}), \mathfrak{b}(\eta_\mathrm{i})\}}$$



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

where $\mathfrak{a}(\eta_i)$ represents the average intra cluster distance, or the average distance between sensor node $(\eta_i)$ and every other sensor node within the same cluster. $SC(\eta_i)$ denotes the silhouette coefficient of the sensor node $(\eta_i)$. $\mathfrak{b}(\eta_i)$ signifies the minimum average inter-cluster gap between sensor point $(\eta_i)$ and all clusters to which $(\eta_i)$ does not belong.The value of the SCranges from $-1$ to 1. A sensor node with a score of 1 is well-separated from other clusters and highly compact within its own cluster. Conversely, $-1$ is the lowest possible value. Scores near 0 indicate overlapping clusters.

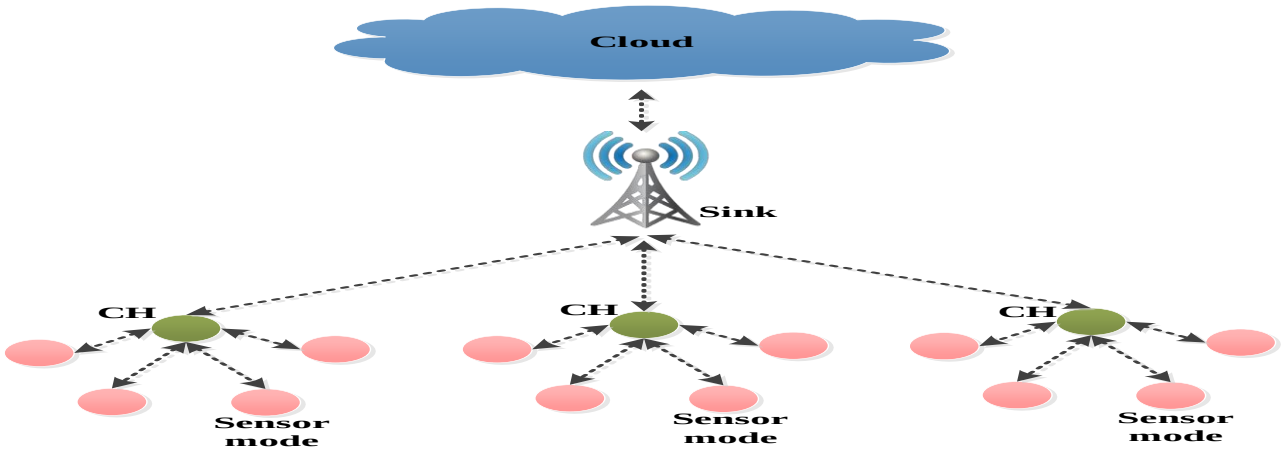


**Figure 2.** IFA based Clustering

**FCM clustering:**

In this section, we present a centralized clustering algorithm that leverages the fuzzy $c$-means (FCM) as shown in figure 2, technique to partition the network into an optimal number of clusters. We assume that the network architecture is fully known to the base station(BS) node, and that each cluster head (CH) maintains a connection to the BS node. Using the information received from the sensor nodes (SN), the BS applies the FCM technique to determine the cluster centroids and allocate SN to clusters $\mathfrak{c} = \{\mathbb{C}_1, \mathbb{C}_2, .., \mathbb{C}_\mathfrak{c}\}$. Instead of assigning nodes to a single cluster, each node is given a degree of membership $\mathcal{Z}_{ij}$ to each cluster $\mathbb{C}_i$. The objective of the iterative FCM process is to locally minimize the following objective function.

$$\Im_{min} = \sum_{i=1}^{\mathfrak{c}} \sum_{j=1}^{n} \mathcal{Z}_{ij}^{m} d_{ij}^{2}$$

where $d_{ij}$ is the distance between sensor node $SN_j$ and the centre of cluster $\mathbb{C}_i$ and $\mathcal{Z}_{ij}$represents the degree of membership of sensor node $SN_j$ to cluster $\mathbb{C}_i$. When $m$ is greater than 1, the number of sensor nodes and the number of clusters $\mathfrak{c}$ together influence the behaviour of the FCM-based clustering method. The clustering phase involves the following steps:

1. Change the number of fuzzy clusters to $\mathfrak{c}$.
2. Randomly choose $\mathfrak{c}$ for the first cluster centres.
3. Determine the membership matrix using.

$$\mathcal{Z}_{ij} = \frac{1}{\sum_{k=1}^{\mathfrak{c}} \left(\frac{d_{ij}}{d_{kj}}\right)^{\frac{2}{m-1}}}$$

4. Utilize the membership matrix to calculate the cluster centres.

$$\mathbb{c}_j = \frac{\sum_{i=1}^{n} \mathcal{Z}_{ij}^{m} \eta_i}{\sum_{i=1}^{n} \mathcal{Z}_{ij}^{m}}$$

5. Repeat steps 3 and 4 until the membership values of all nodes converge. This process can be expressed as follows: $max\{\mathcal{Z}_{ij}^{I+1} - \mathcal{Z}_{ij}^{I}\} < \gamma$



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

Where $\gamma$ is the termination criterion, ranging from 0 to 1, and $\mathcal{Z}_{ij}^{I}$ represents the membership matrix at the $I$-th iteration. Once $\mathcal{Z}_{ij}^{I}$ converges, each node is assigned to the cluster where it has the highest membership.

### B. Efficient Task Classification and Scheduling

Tasks from IoT devices are classified using a Hybrid Deep Learning (HDL) algorithm, where LiteCNN categorizes them into emergency and non-emergency tasks, and LiteLSTM schedules them to minimize the weight matrix and overfitting. Dual scheduling based on task length, delay sensitivity, QoS, priority, resource consumption, and queue length further reduces execution time and latency.

| | Latency, processing Time, total execution time (deadline) (seconds) | | | |
|---|---|---|---|---|
| **Request** | **Edge server** | **Cloud server 1** | **Cloud server 2** | **Cloud server 3** |
| 1 | (0.46, 8.34, 9.27) | (0.56, 3.15, 4.11) | (0.56, 2.88, 3.86) | (0.64, 2.55, 3.52) |
| 2 | (0.48, 9.16, 10.62) | (0.61, 3.46, 4.93) | (0.61, 5.54, 7.03) | (0.70, 2.80, 4.30) |
| 3 | (0.45, 2.15, 3.89) | (0.58, 0.88, 2.63) | (0.57, 2.93, 4.95) | (0.65, 3.32, 5.10) |
| 4 | (0.46, 9.21, 11.23) | (0.59, 3.46, 5.50) | (0.59, 5.85, 7.88) | (0.67, 5.07, 7.11) |
| 5 | (0.44, 6.68, 8.91) | (0.56, 5.74, 7.98) | (0.56, 4.77, 7.01) | (0.64, 4.12, 6.37) |
| | **Energy Consumption (watt-seconds)** | | | |
| **Request** | **Edge server** | **Cloud server 1** | **Cloud server 2** | **Cloud server 3** |
| 1 | 1428.689702 | 3686.826824 | 471.5617768 | 1144.184523 |
| 2 | 1602.387358 | 4187.575242 | 688.1995320 | 1321.858751 |
| 3 | 313.592157 | 488.4034943 | 71.33667920 | 154.8914872 |
| 4 | 1614.936248 | 4234.461695 | 699.321682 | 1334.983995 |
| 5 | 1103.267920 | 2594.204827 | 364.7407357 | 821.1816278 |

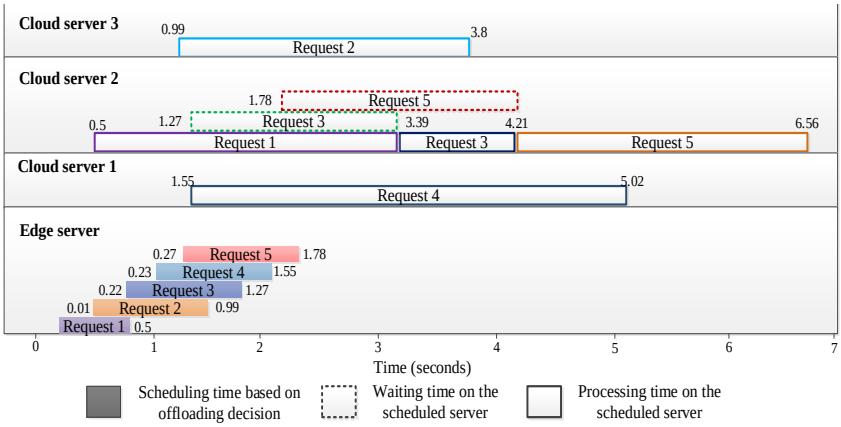


**Figure 3.** Task Scheduling

Instead of building the model from scratch, the weights for the VGG16 layers were taken from the pre-trained VGG16 model on ImageNet, which contains millions of images. This allows for the extraction of features using highly optimized weights. Figure 4 illustrates an example of features retrieved from several convolution layers, showing that deeper layers capture more precise details from the image. To ensure adequate learning capacity without overfitting, a robust architecture is essential. Consequently, after the fourth convolution block, a batch normalization layer is introduced to normalize the layer output and prevent overfitting. Batch normalization can be beneficial to any layer of the neural network. It is primarily used to create a stable distribution of activation values, which reduces internal covariate shift and helps to prevent overfitting [16]. For each dimension of the $\mathbb{d}$-dimensional input, $\mathfrak{y} = \left(\mathfrak{y}^{(1)} \ldots \mathfrak{y}^{(\mathbb{d})}\right)$, the normalization process is applied by

$$\hat{\mathfrak{y}}^{k} = \frac{\mathfrak{y}^{(k)} - \mathbb{E}\left[\mathfrak{y}^{(k)}\right]}{\sqrt{Var[\mathfrak{y}^{(k)}]}}$$



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

where $\mathfrak{y}^{(k)}$ represents each individual activation, $\mathbb{E}[\mathfrak{y}^{(k)}] = \frac{1}{m}\sum_{i=1}^{m}\mathfrak{y}_i^{(k)}$, $Var[\mathfrak{y}^{(k)}] = \frac{1}{m}\sum_{i=1}^{m}(\mathfrak{y}_i^{(k)} - \mathbb{E}[\mathfrak{y}^{(k)}])^2 + \mathfrak{e}$denote the mean and variance, with $\mathfrak{e}$ being a constant added for numerical stability. The normalized value is typically scaled by $\mathfrak{s}^k$ and offset by $\mathfrak{o}^k$ to ensure the input value is not constrained to a small range:

$$\mathfrak{x}^{(k)} = \mathfrak{s}^k\hat{\mathfrak{y}}^k + \mathfrak{o}^k$$

Normally, the final three fully connected layers of the VGG16 architecture are used for classification; however, in this instance, they are omitted. The mathematical representation of the standard LSTM structure is provided by:

$$i_t = sig(\mathcal{W}_{ii}\mathcal{A}_t + \mathcal{W}_{hi}\mathcal{B}_{t-1} + \mathbb{b}_i)$$
$$d_t = sig(\mathcal{W}_{if}\mathcal{A}_t + \mathcal{W}_{hf}\mathcal{B}_{t-1} + \mathbb{b}_f)$$
$$\mathfrak{g}_t = tanh(\mathcal{W}_{ig}\mathcal{A}_t + \mathcal{W}_{hg}\mathcal{B}_{t-1} + \mathbb{b}_g)$$
$$o_t = sig(\mathcal{W}_{io}\mathcal{A}_t + \mathcal{W}_{ho}\mathcal{B}_{t-1} + \mathbb{b}_o)$$
$$\mathcal{C}_t = d_t * \mathcal{C}_{t-1} + i_t * \mathfrak{g}_t$$
$$\mathbb{Q}_t = o_t * tanh(\mathcal{C}_t)$$

where the sigmoid function $sig$ is an element-wise logistic sigmoid activation function;$\mathcal{W}$ and $\mathbb{b}$ represent the weight matrices and bias vectors, respectively; $\mathcal{A}_t$ and $\mathcal{B}_t$ denote the input and hidden states at time step $t$; $*$ indicates element-wise multiplication; $i_t$, $d_t$, and $o_t$ represent the input gate, forget gate, and output gate, respectively; and $\mathcal{C}_t$ and $\mathfrak{g}_t$ denote the cell state and hidden vector, respectively.

$$\begin{bmatrix} i_t \\ d_t \\ \mathfrak{g}_t \\ o_t \end{bmatrix} = \begin{bmatrix} sig \\ sig \\ tanh \\ sig \end{bmatrix} * (\mathcal{W}_i\mathcal{A}_t + \mathcal{W}_h\mathcal{B}_{t-1} + \mathbb{B})$$

$$\mathcal{W}_i = [\mathcal{W}_{ii}, \mathcal{W}_{if}, \mathcal{W}_{ig}, \mathcal{W}_{io}]^{\mathbb{T}}$$
$$\mathcal{W}_h = [\mathcal{W}_{hi}, \mathcal{W}_{hf}, \mathcal{W}_{hg}, \mathcal{W}_{ho}]^{\mathbb{T}}$$
$$\mathbb{B} = [\mathbb{b}_i, \mathbb{b}_f, \mathbb{b}_g, \mathbb{b}_o]^{\mathbb{T}}$$

For IoT devices, the traditional LSTM model is often too large or complex. In typical LSTM models, most of the computational complexity arises from matrix-vector multiplications, specifically $\mathcal{W}_i\mathcal{A}_t$ and $\mathcal{W}_h\mathcal{B}_{t-1}$. By using dimensionality reduction and singular value decomposition (SVD), we can reduce the computational complexity of these matrix-vector multiplications. Suppose we have a matrix $\mathcal{W} \in R^{\mathfrak{m}\times\mathfrak{n}}$. Using SVD, we can decompose $\mathcal{W}$ into $\mathcal{X} \in R^{\mathfrak{m}\times\mathfrak{n}}$, $\mathcal{Y} \in R^{\mathfrak{m}\times\mathfrak{m}}$, and $\mathfrak{Z} \in R^{\mathfrak{n}\times\mathfrak{n}}$. One approach to reducing the dimensionality of $\mathcal{X}$, $\mathcal{Y}$, and $\mathfrak{Z}$ is to set all small singular values to zero. By doing so, the corresponding $s$ columns of $\mathcal{X}$ and $\mathcal{Y}$ can also be removed if the $s$ small singular values are set to zero.

$$\mathcal{W}_{\mathfrak{m}\times\mathfrak{n}} = \mathcal{X}_{\mathfrak{m}\times\mathfrak{m}}\mathcal{Y}_{\mathfrak{m}\times\mathfrak{n}}\mathfrak{Z}^{\mathbb{T}}_{\mathfrak{n}\times\mathfrak{n}}$$
$$\approx \mathcal{X}_{\mathfrak{m}\times\mathbb{r}}\mathcal{Y}_{\mathbb{r}\times\mathbb{r}}\mathfrak{Z}^{\mathbb{T}}_{\mathbb{r}\times\mathfrak{n}}$$
$$= \mathcal{X}_{\mathfrak{m}\times\mathbb{r}}\mathbb{N}_{\mathbb{r}\times\mathfrak{n}}$$

Additionally, the energy in $\mathfrak{Z}$ is determined by the number, $\mathbb{r}$ of singular values. The proposed lightweight LSTM model involves decomposing $\mathcal{W}_h$ into two matrices, $\mathcal{W}_{h1}$ and $\mathcal{W}_{h2}$. $\mathcal{W}_h$ has dimensions of $\mathfrak{m} \times \mathfrak{n}$, $\mathcal{W}_{h1}$ has dimensions of $\mathfrak{m} \times \mathbb{r}$, and $\mathcal{W}_{h2}$ has dimensions of $\mathbb{r} \times \mathfrak{n}$. Consequently, in the proposed LSTM, we only encounter $(\mathfrak{m} \times \mathbb{r} + \mathbb{r} \times \mathfrak{n})$ weights in $\mathcal{W}_{h1}$ and $\mathcal{W}_{h2}$, whereas in the standard LSTM, we would find $\mathfrak{m} \times \mathfrak{n}$ weights in $\mathcal{W}_h$. We similarly decompose $\mathcal{W}_i$ into two matrices, $\mathcal{W}_{i1}$ and $\mathcal{W}_{i2}$, as we did with $\mathcal{W}_h$. According to equation (10), the traditional LSTM cell involves$8.(\mathfrak{m} + \mathfrak{n})\mathbb{r}$ matrix–vector multiplications. Therefore, the ratio of multiplications in the suggested LSTM cell to those in the traditional LSTM cell, also known as the compression ratio, is as follows.

$$CR = \frac{(\mathfrak{m} + \mathfrak{n})\mathbb{r}}{\mathfrak{m}\mathfrak{n}}$$

Applying dimensionality reduction to the matrices $\mathcal{W}_h$ and $\mathcal{W}_i$ of the standard LSTM model, Figure 3 proposes a lightweight LSTM structure. For instance, if $\mathcal{W}_h$ is of size $5 \times 20$, we calculate $5 \times 20 = 100$ weights for $\mathcal{W}_h$. The weight matrix can be



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

approximately represented as $\mathcal{W}_h \approx \mathcal{W}_{h1}\mathcal{W}_{h2}$, and with a reduction in dimensionality setting $\mathbb{r} = 2$ . the dimensions of $\mathcal{W}_{h1}$ and $\mathcal{W}_{h2}$ become $5 \times 2$ and $2 \times 20$, respectively. Therefore, we only need to find 50 weights. In other words, by reducing the dimensionality, we can halve the complexity of the LSTM model.

### C. Trading based Allocation

The traits that were extracted represent the state denoted by F. The objective is defined in two dimensions, incorporating both the ground truth error and detection. The first dimension of F facilitates measuring the distance between the actual and detected tumor locations. The formulation is provided below.

$$\mathrm{Dis_t} = \min\left(\frac{\left\|\mathbb{DT}_{\mathrm{t}}^{<x,y>} - \mathbb{GT}_{\mathrm{t}}^{<x,y>}\right\|_2}{\Xi}, 1.0\right)$$

In the preceding equation, the ground truth position of the colon tumor is represented by $\mathbb{GT}_{\mathrm{t}}^{<x,y>}$, while the detected tumor location in the first dimension is indicated by $\mathbb{DT}_{\mathrm{t}}^{<x,y>}$. The normalizing constant $\Xi$ is used to map the error distance within the interval $[0,1]$. The Euclidean norm operation is represented by the symbols $\|.\|_2$. The ground boundary errors and detected tumor for the colon are provided in the second dimension. The formulation is as follows:

$$\Delta\Xi_t = a\tan\left(\left(\mathbb{G}T^{<Y>} - \mathbb{D}T_t^{<y>}\right), \left(\mathbb{G}T^{<x>} - \mathbb{D}T_t^{<x>}\right)\right) - \Xi_t$$

The value of $\Xi_t$ in the equation above specifies the boundary of the colon tumor. To define the normalization of the tumor border error, one method is as follows:

$$\Delta\Xi_{\mathrm{t}} = \begin{cases} \dfrac{\Delta\Xi_{\mathrm{t}} + 2\rho}{\rho}, & \text{if}\Xi_{\mathrm{t}} < -2\rho \\ \dfrac{\Delta\Xi_{\mathrm{t}} - 2\rho}{\rho}, & \text{if}\Xi_{\mathrm{t}} > 2\rho \\ \dfrac{\Delta\Xi_{\mathrm{t}}}{\rho}, & \text{Else} \end{cases}$$

The adopted MA-DRL enabled decision maker uses the previously indicated boundary inputs to generate instructions in the form of actions, denoted as $\mathrm{act_t} = [\mathrm{Local_{err}}, \mathrm{Bound_{err}}]$, where $\mathrm{Local_{err}}$ and $\mathrm{Bound_{err}}$ represent the localization error and boundary error, respectively. The primary goal of the MADRL-aided decision is to ensure accurate tumor categorization. To enhance convergence efficiency, the reward function has been carefully designed. To make it more understandable, the reward function can be expressed as follows:

$$\mathrm{rew_t} = \mathrm{rew_{FP}} + \mathrm{rew_{act}} + \mathrm{rew_{bound}}$$

In the equation above, the boundary error reward is represented by $\mathrm{rew_{bound}}$, the action reward by $\mathrm{rew_{act}}$, and the false positive error reward by $\mathrm{rew_{FP}}$ . Ultimately, the adopted Soft Actor Critic (SAC) intelligently learns the decision policy $\left(\mho(\mathrm{act_t}|\mathrm{rew_t})\right)$based on the extracted features F from the DCA-ECTNet at time stamp $t$.To address the overestimation issue, the SAC algorithm utilized the dual $\mathbb{Q}$-Network approach as a collaborative operation. Consequently, the Mean Bellman-Squared Error (MBSE) loss is employed to robustly reconstruct the MA-DRL-DCAECTNet critic parameters. This can be expressed as follows:

$$\mathrm{Loss}(\theta_{\mathrm{j}}) = \underset{\mathrm{rew_t}\sim\gamma,\mathrm{act_t}\sim\daleth}{\epsilon}\left\|\mathbb{q}_{\theta_{\mathrm{j}}}^{\daleth}(\mathrm{rew_{FP}}, \mathrm{act_t}) - \left(\mathrm{rew_{bound}} + \lambda.\widehat{\mathbb{Q}}^{\daleth}\right)\right\|_2$$

The state-action estimate function for the dual $\mathbb{Q}$ networks is derived from the preceding equation as follows:

$$\widehat{\mathbb{Q}}^{\daleth} = \underset{\mathrm{rew_{t+1}}\sim\gamma,\mathrm{act_{t+1}}\sim\daleth}{\epsilon}\left[\min_{\mathrm{j}=1,2}\mathbb{Q}_{\theta_{\mathrm{j}}}^{\daleth}\mathrm{tar}(\mathrm{rew_{t+1}}, \mathrm{act_{t+1}}) - \eth.\log\daleth\left(\mathrm{act_{t+1}}|\mathrm{rew_{t+1}}\right)\right]$$

The trade-off parameter that ensures a balance between the stochastic policy and the state-action value is represented by the symbol $\eth$ in the equation above. It appears that the actor networks are updated by optimizing the soft state-action functions, which are expressed as follows:

$$Loss(\theta) = \underset{\mathrm{rew_t}\sim\gamma,\mathrm{act_t}\sim\daleth}{\epsilon}\left[\mathbb{q}_{\theta_{\mathrm{j}}}^{\daleth}(\mathrm{rew_{FP}}, \mathrm{act_t}) - \eth.\log\daleth_{\theta}\left(\mathrm{act_{t+1}}|\mathrm{rew_{t+1}}\right)\right]$$

The algorithm fully demonstrates the proposed DCA-ECTNet-based MA-DRL enabled decision-making process.

## 4. Numerical results and discussion



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

### A. Simulation Setup

We envision a 10-kilometer, one-way roadway for the experiment, as illustrated in Figure 3. To provide a more realistic simulation, the route is divided into sections. Additionally, we placed 20 RSUs evenly spaced along the road, with RSUs 1 and 20 located at the far ends of the segments to the left and far right, respectively. We anticipate that all vehicles will enter the road from the far-left side, within the RSU1 coverage region. During the simulation, we considered four distinct vehicle speeds: low (30 km/h), medium (60 km/h), high (100 km/h), and extremely high (200 km/h). Typically, a nearby RSU receives the computed tasks of the vehicles. We assume that each RSU has a coverage area of 500 meters and an access point with IEEE 802.11p capability. Each RSU is equipped with a single MEC server hosting four virtual machines (VMs). For the network delay model, we utilize the Markov modulated process (MMPP/M/1 queue model). In the context of V2R communication, we use an IEEE 802.11p protocol interface with a data rate of 10 Mbps to transmit the computed tasks. Our system can also transfer a vehicle's tasks to a cloud server via an RSU or an LTE base station. During the simulation analysis, we used the LTE BS following the measurements provided. Additionally, Table 3 lists other parameters, including job duration, maximum delay allowed, and upload and download data sizes for various applications.

### B. Performance Analysis

To evaluate the performance of our proposed OTO model based on diverse scenarios by EdgeCoudSim simulator, we compared our proposed in two approaches: (i) with three traditional offloading schemes and (ii) several existing works for proving our model's efficacy.

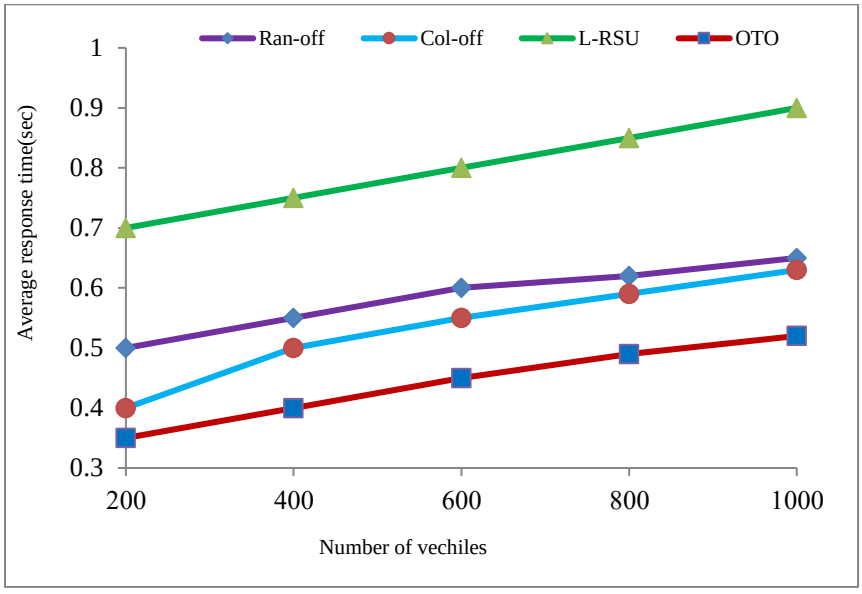


**Figure 4.** Average Response Time

Figure 4 shows the average response times (y-axis) versus the number of cars (x-axis, from 100 to 1000) to assess the OTO technique. As the number of vehicles increases, the average response time rises for all methods. The LRC scheme has a longer response time due to congestion, while other methods distribute tasks between the remote cloud and edge, preventing a similar increase. The OTO strategy achieves lower response times by adjusting task-offloading probability, reducing response time by 47.6%, 32.6%, and 26.5% compared to LRC, random, and collaborative methods. Figure 5 shows that the task-failure rate remains low up to 500 vehicles but rises with more congestion. Figure 5 shows that the OTO system can manage 1000 cars without congestion, while the LRC scheme faces congestion after 400 vehicles, and the random and collaborative offloading schemes after 600 vehicles. The LRC approach gets congested due to limited MEC server capacity. In contrast, the random and collaborative methods handle up to 600 vehicles by distributing tasks between the edge and the cloud, but then experience congestion due to WAN delays. The random offloading strategy also transfers some larger tasks to MEC servers, causing longer completion times.



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

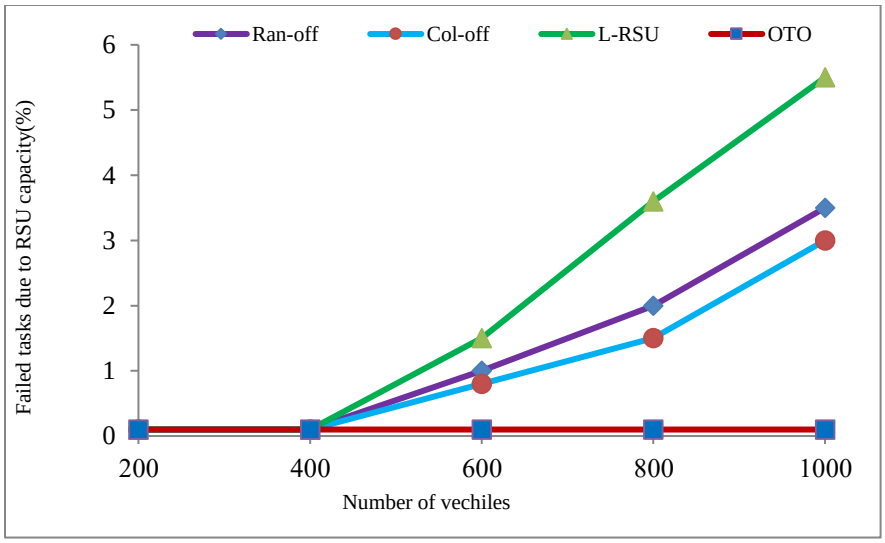


**Figure 5.** Failed Task Due to RSU Capacity

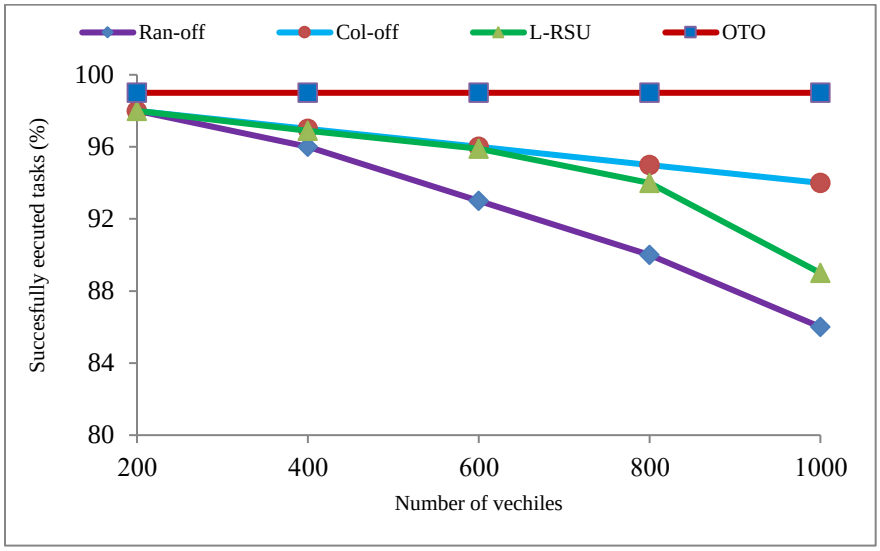


**Figure 6.** Effectively Completed Assignments for Different Quantities of Vehicles

OTO strategy effectively uses MEC server resources and dynamically offloads tasks between MEC and the cloud, maintaining a stable equilibrium to handle 1000 cars without congestion. Figure 6 highlights task completion rates for different vehicle counts. Under light loads, all schemes successfully complete most offloaded jobs. As vehicle numbers increase, completion rates differ: the LRC scheme's success rate drops from 99.5% at 500 vehicles to below 91.4% at 1000 vehicles due to MEC server overload. The random offloading strategy's success rate decreases from 99.6% to 85.4%. The OTO strategy shows a minor decline from 99.8% to 99.5%. The collaborative offloading scheme drops from 99.7% to 95.2% due to WAN delays. This demonstrates the OTO system's effectiveness in managing higher vehicle counts. Figure 6 shows that the OTO strategy completes more offloaded tasks than other methods by making better offloading decisions and efficiently using MEC resources. An additional experiment, shown in Figure 7, analyzed average response times by varying the ratio of latency-tolerant infotainment programs to latency-sensitive hazard assessment applications. Ratios ranged from 0:10 to 10:0 (e.g., 7:3 means seven infotainment apps for every three hazard assessment apps). At a 0:10 ratio, the average response times for LRC, random, collaborative, and OTO systems were 2.11, 1.43, 1.13, and 0.69 seconds, respectively. Larger infotainment tasks caused longer response times, but reducing their proportion decreased response times. The OTO system consistently outperformed others, maintaining low response times across various task ratios without exceeding deadlines.



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

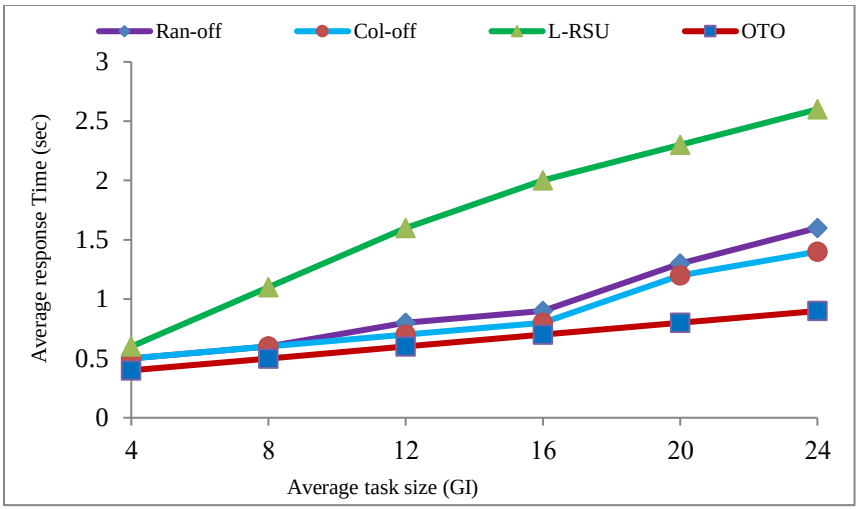


**Figure 7.** Average Response Time across Different Ratios of Latency-Sensitive to Latency-tolerant Tasks

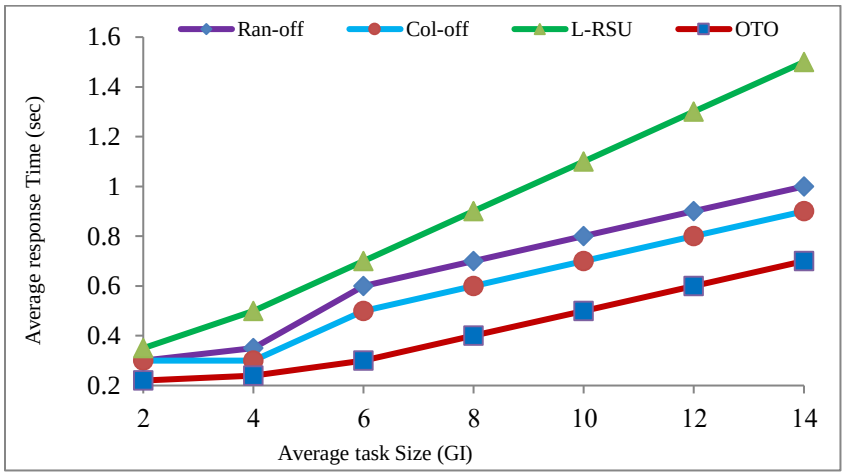


**Figure 8.** Average Response Times across Various Task Sizes within Infotainment Applications

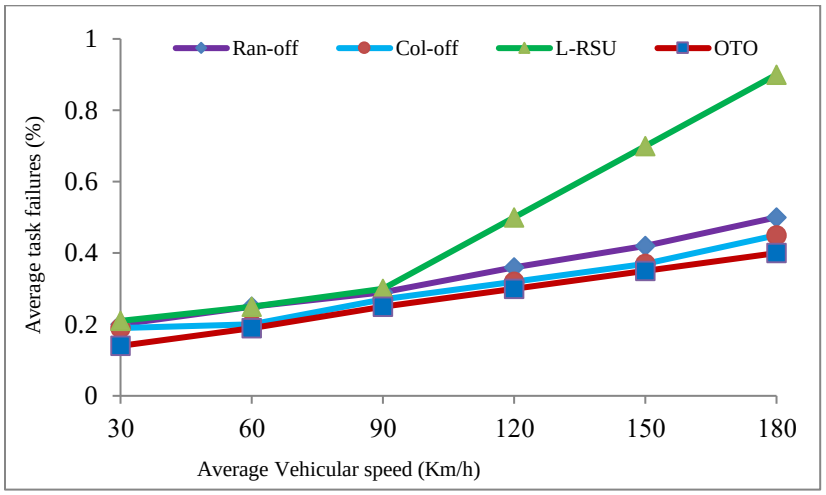


**Figure 9.** Average Response Times across Various Task Sizes within Danger Assessment Applications



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

We conducted further tests, depicted in Figures 8 and 9, to examine the effects of different task sizes for infotainment and risk assessment applications. Analysis of these figures revealed that the average response time across all offloading schemes is nearly identical when the average job size is small. However, the response time increases with larger task sizes. The infotainment application has a slower average response time compared to the navigation application due to generating larger tasks. From this analysis, we conclude that our OTO solution performs better than others in every scenario. The OTO scheme dynamically processes offloaded tasks and converges to a unique equilibrium. The final simulation result, shown in Figure 10, examines the impact of different vehicle speeds on job failure rates. We used 500 cars with average speeds ranging from 30 to 180 km/h.

**(ii) Existing Task Offloading Schemes**

Figure 10 shows the energy consumption of vehicles using DR-ACO [21], ADL-CO [23], E2-CO [25], E-AGA [30] and OTO. OTO consumes the least energy due to its optimization strategy, which minimizes energy use for each request. E2-CO [25], which schedules requests only on edge servers and not the cloud, uses more energy and takes longer to process requests, increasing overall energy consumption. DR-ACO, although not highly efficient, uses less energy than E2-COby handling requests via the cloud, reducing both time and energy usage. ADL-CO and E-AGA consumes the most energy because it schedules requests randomly without considering SLAs, resulting in longer execution times and higher energy usage.Figure 11 displays the latencies of queries made with DR-ACO, ADL-CO, E2-CO, E-AGAand OTO, showing that all algorithms have latencies below the 500-millisecond maximum allowable latency. E2-CO has the shortest request latencies because it only considers the edge layer, unlike OTO, DR-ACO, and ADL-CO, which also incorporate the cloud layer, resulting in higher latencies.

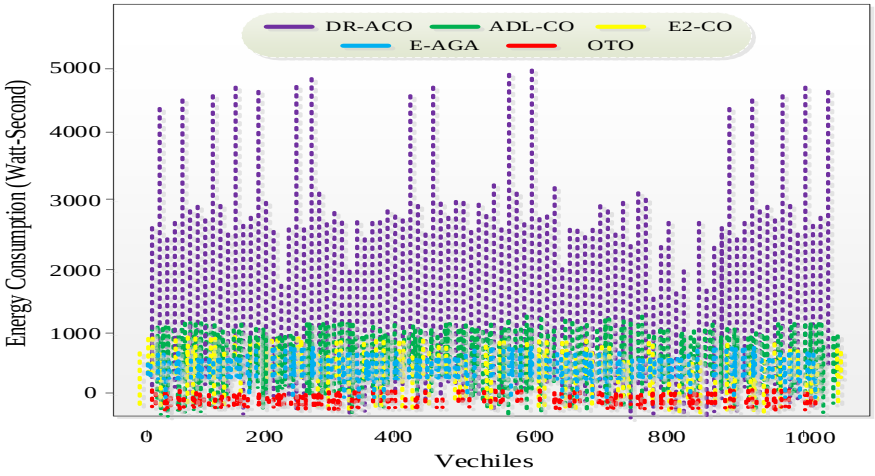


**Figure 10.** Performance Analysis of Energy Consumption

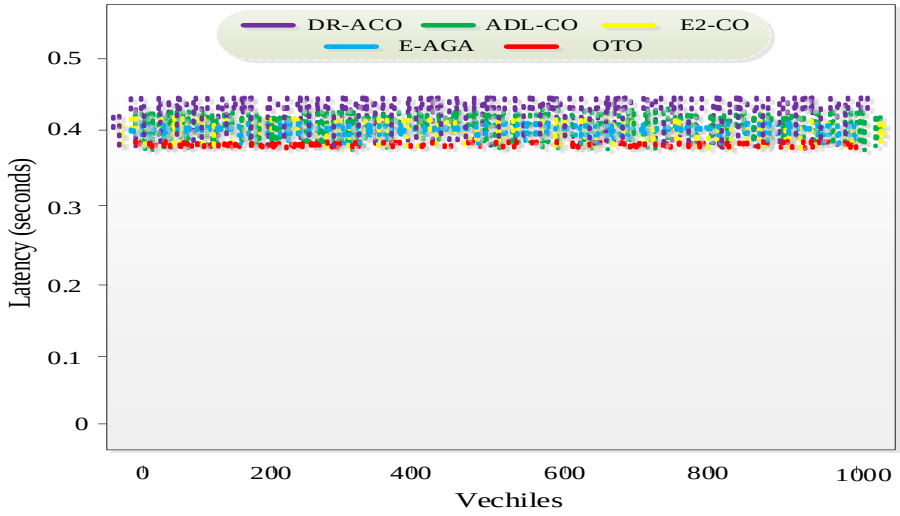


**Figure 11.** Performance Analysis of Latency



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

Figure 12 shows the processing times of requests using DR-ACO, E2-CO, ADL-CO, E-AGA and OTO. All requests processed with DR-ACO and OTO have processing times under the 5-second maximum due to their consideration of processing time constraints during scheduling. A zoomed-in view of DR-ACO and OTO processing times is provided for clarity. Not all requests for E2-CO and ADL-CO meet the processing time limit; ADL-CO doesn't account for this need, and E2-CO relies solely on edge servers, which can't always meet the requirement. As waiting times for requests increase with more cars, so does processing time in E2-CO and E-AGA. Figure 13 displays the total execution time of requests made with DR-ACO, E2-CO, ADL-CO, E-AGA and OTO. All queries using DR-ACO and OTO have execution times within the permitted window, due to their consideration of deadline constraints during scheduling. An enlarged view of DR-ACO and OTO execution timings is included for better understanding. E2-CO and ADL-CO fail to meet the deadline constraint because E2-CO's edge servers can't handle requests in time, and ADL-CO ignores the deadline constraint. As the number of vehicles increases, execution times for E2-COand E-AGA also increase due to longer wait times at the edge servers.

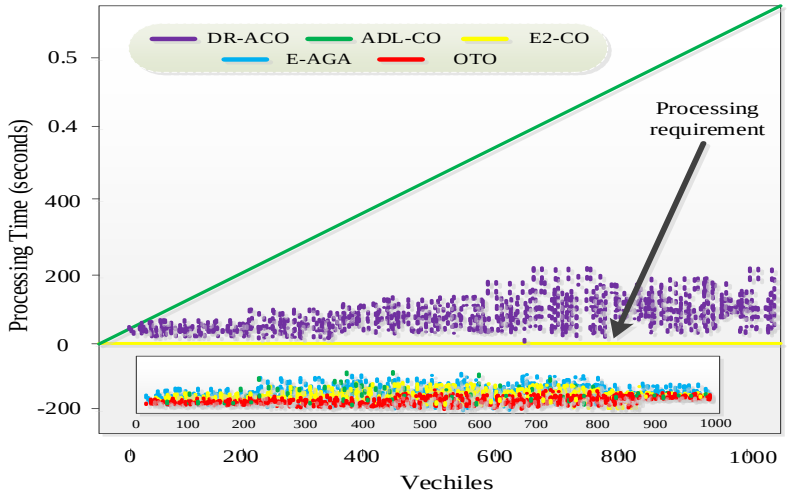


**Figure 12.** Processing Request Time

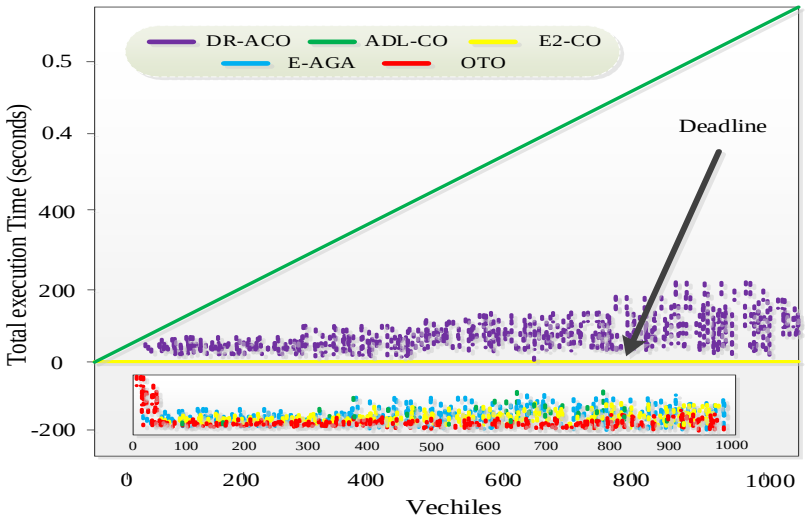


**Figure 13.** Total Execution Request Time

Figure 14 shows the fraction of total energy consumption using DR-ACO, E2-CO, ADL-CO, E-AGA and OTO, along with the overall energy usage of all requests. OTO has the lowest energy consumption with 0% SLAVs. DR-ACO also has 0% SLAVs but consumes more energy than OTO. E2-CO uses more energy than OTO and DR-ACO combined, with an SLAV percentage of 99.5%. ADL-CO and E-AGA performs the worst in terms of energy usage, with 92.4% of SLAVs using ADL-CO. Figure 15 illustrates the average latency, processing, and overall execution times of the algorithms. All algorithms have average latencies below the allowable limit. OTO and DR-ACO meet the processing and total execution time limits, while E2-CO and ADL-CO do



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

not meet deadlines and processing time requirements. In summary, OTO saves 60.26% of total energy consumption compared to E2-CO, 1.47% compared to DR-ACO, and 69.44% compared to ADL-CO, with no SLAVs.

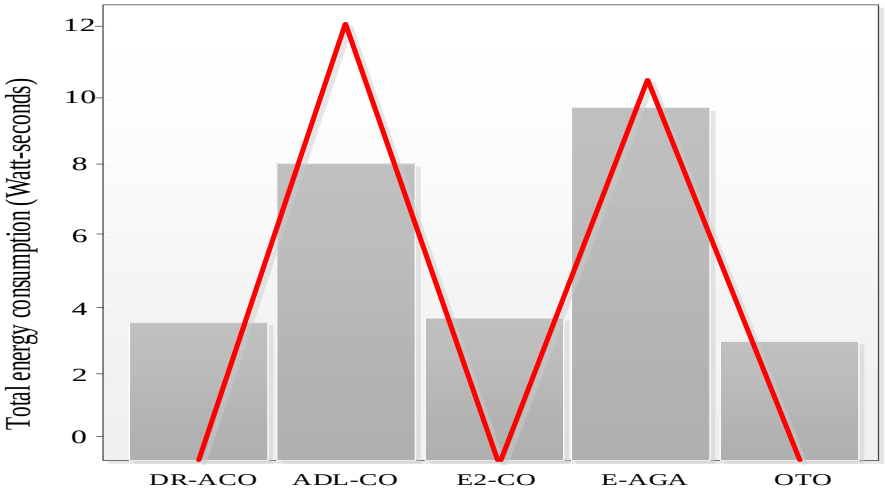


**Figure 14.** Total Energy Consumption

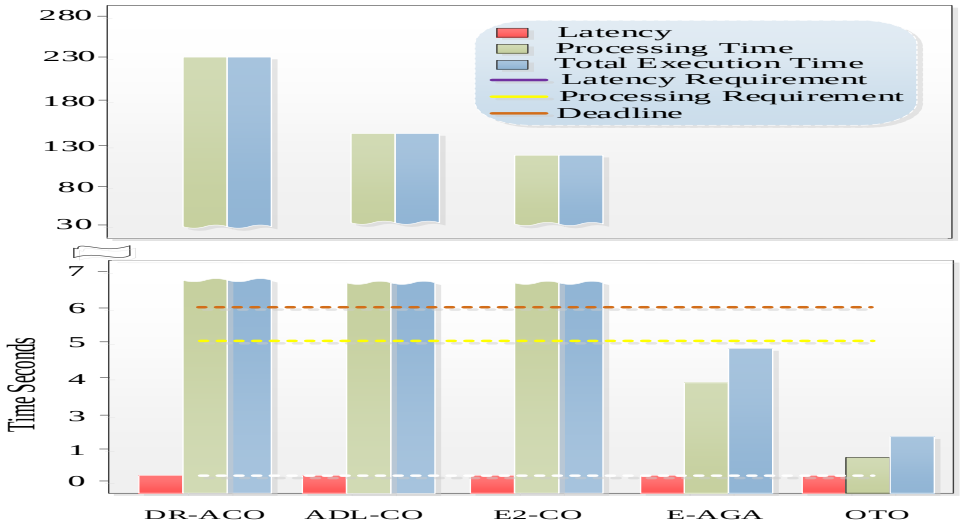


**Figure 15.** Average Processing, Total Execution Time and Latency

## 6. Conclusion

In recent years, VANETs have emerged as a pivotal technology for enabling efficient and reliable communication among vehicles and infrastructure. The dynamic and mobile nature of VANETs presents unique challenges in terms of task scheduling, resource allocation, and latency management. In this paper, we introduced the OTO framework designed specifically to address these challenges and enhance the performance of VANETs.The primary goal of the OTO framework is to ensure timely and efficient task execution by leveraging advanced algorithms and techniques tailored to the characteristics of vehicular networks. Key innovations include the use of the IFA for clustering UEs based on proximity, and signal strength. This initial clustering step is crucial as it forms the foundation for minimizing latency and conserving energy through optimized UE groupings.Moreover, the framework incorporates an HDL approach consisting of LiteCNN and LiteLSTM models. LiteCNN is employed for real-time classification of tasks into emergency and non-emergency categories, while LiteLSTM handles the dynamic scheduling of these tasks based on factors such as urgency, QoS requirements, and available resources. This dual-model approach not only reduces the complexity associated with traditional task management but also improves accuracy and responsiveness in decision-making processes within the network.Central to the OTO framework is its innovative approach to edge server allocation using a RL-based MA-DRL algorithm. This approach treats edge servers as sellers and UEs as buyers, optimizing task distribution based on load balancing,



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

resource availability, waiting time, and geographical proximity. By employing RL techniques, the framework adapts dynamically to changing network conditions and user demands, thereby mitigating latency spikes and enhancing overall network performance.Throughout our comparative analysis, we rigorously evaluated the OTO framework against existing methodologies across various performance metrics. Results consistently demonstrated that the proposed framework achieves significant improvements in latency reduction, resource utilization efficiency, and overall system performance.

**References**


**[1]** Gupta, B. B., Gaurav, A., Marín, E. C., & Alhalabi, W. (2022). Novel graph-based machine learning technique to secure smart vehicles in intelligent transportation systems. *IEEE transactions on intelligent transportation systems*.

**[2]** Kumar, N., Poonia, V., Gupta, B. B., & Goyal, M. K. (2021). A novel framework for risk assessment and resilience of critical infrastructure towards climate change. *Technological Forecasting and Social Change*, *165*, 120532.

**[3]** Devi Murugavel, Kiruthiga, Parthasarathy Ramadass, Rakesh Kumar Mahendran, Arfat Ahmad Khan, Mohd Anul Haq, Sultan Alharby, and Ahmed Alhussen. 2022. "Maintaining Effective Node Chain Connectivity in the Network with Transmission Power of Self-Arranged AdHoc Routing in Cluster Scenario" Electronics 11, no. 15: 2455. https://doi.org/10.3390/electronics11152455.

**[4]** Ruphitha, S.V., Ambeth Kumar, V.D., " Predictive analysis of postpartum haemorrhage using deep learning technique", Advances in Parallel Computing, 2021, 38, pp. 168–172.

**[5]** Liu, Z., Weng, J., Guo, J., Ma, J., Huang, F., Sun, H., & Cheng, Y. (2021). PPTM: A privacy-preserving trust management scheme for emergency message dissemination in space–air–ground-integrated vehicular networks. *IEEE Internet of Things Journal*, *9*(8), 5943-5956.

**[6]** Avasalcai, C., Zarrin, B., &Dustdar, S. (2021). EdgeFlow—Developing and deploying latency-sensitive IoT edge applications. *IEEE Internet of Things Journal*, *9*(5), 3877-3888.

**[7]** Iqbal, U., Tandon, A., Gupta, S., Yadav, A. R., Neware, R., &Gelana, F. W. (2022). A novel secure authentication protocol for IoT and cloud servers. *Wireless Communications and Mobile Computing*, *2022*(1), 7707543.

**[8]** R. K. Mahendran, S. Rajendran, P. Pandian, R. S. Rathore, F. Benedetto and R. H. Jhaveri, "A Novel Constructive Unceasement Conditional Random Field and Dynamic Bayesian Network Model for Attack Prediction on Internet of Vehicle," in IEEE Access, vol. 12, pp. 24644-24658, 2024, doi: 10.1109/ACCESS.2024.3363420.

**[9]** Zhang, Y., Zhang, H., Zhou, H., Long, K., & Karagiannidis, G. K. (2022). Resource allocation in terrestrial-satellite-based next generation multiple access networks with interference cooperation. *IEEE Journal on Selected Areas in Communications*, *40*(4), 1210-1221.

**[10]** Indhumathi, M., Ambeth Kumar, V.D, " Future prediction of cardiovascular disease using deep learning technique", Advances in Parallel Computing, 2021, 38, pp. 219–223.

**[11]** Mohtavipour, S. M., Saeidi, M., &Arabsorkhi, A. (2022). A multi-stream CNN for deep violence detection in video sequences using handcrafted features. *The Visual Computer*, *38*(6), 2057-2072.

**[12]** Bute, M. S., Fan, P., Zhang, L., & Abbas, F. (2021). An efficient distributed task offloading scheme for vehicular edge computing networks. *IEEE Transactions on Vehicular Technology*, *70*(12), 13149-13161.

**[13]** Dai, X., Xiao, Z., Jiang, H., & Lui, J. C. (2023). UAV-assisted task offloading in vehicular edge computing networks. *IEEE Transactions on Mobile Computing*.

**[14]** Zeng, F., Chen, Y., Yao, L., & Wu, J. (2021). A novel reputation incentive mechanism and game theory analysis for service caching in software-defined vehicle edge computing. *Peer-to-Peer Networking and Applications*, *14*(2), 467-481.

**[15]** Wu, L., Zhang, R., Li, Q., Ma, C., & Shi, X. (2022). A mobile edge computing-based applications execution framework for Internet of Vehicles. *Frontiers of Computer Science*, *16*(5), 165506.

**[16]** Rostami, M., Berahmand, K., & Forouzandeh, S. (2021). A novel community detection based genetic algorithm for feature selection. *Journal of Big Data*, *8*(1), 2.

**[17]** Anagnostaki, A. P., Pavlopoulos, S., Kyriakou, E., &Koutsouris, D. (2002). A novel codification scheme based on the" VITAL" and" DICOM" standards for telemedicine applications. *IEEE Transactions on Biomedical Engineering*, *49*(12), 1399-1411.

**[18]** T.S. Shanthi, L. Dheepanbalaji, R. Priya, V.D. Ambeth Kumar, Abhishek Kumar, P. Sindhu, Ankit Kumar, Illegal fishing, anomalous vessel behavior detection through automatic identification system, Materials Today: Proceedings, Volume 62, Part 7, 2022, Pages 4685-4690, 2022

**[19]** R. K. Mahendran, A. Thiyagarajan, A. G. A and K. P, "Multi-Modal Visual Features Perception Technology for Internet of Vehicles (IoV)," 2024 International Conference on Emerging Smart Computing and Informatics (ESCI), Pune, India, 2024, pp. 1-5, doi: 10.1109/ESCI59607.2024.10497246.

**[20]** Ye, J., Zhan, J., & Xu, Z. (2020). A novel decision-making approach based on three-way decisions in fuzzy information systems. *Information Sciences*, *541*, 362-390.




DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024

**[21]** Sherubha, "Graph Based Event Measurement for Analyzing Distributed Anomalies in Sensor Networks", Sådhanå(Springer), 45:212, https://doi.org/10.1007/s12046-020-01451-w

**[22]** Piyush K. Pareek, Pixel Level Image Fusion in Moving objection Detection and Tracking with Machine Learning ",Fusion: Practice and Applications, Volume 2 , Issue 1 , PP: 42-60, 2020

**[23]** Shivam Grover, Kshitij Sidana, Vanita Jain, "Egocentric Performance Capture: A Review", Fusion: Practice and Applications, Volume 2, Issue 2 , PP: 64-73, 2020.

**[24]** Abdel Nasser H. Zaied, Mahmoud Ismail and Salwa El- Sayed, A Survey on Meta-heuristic Algorithms for Global Optimization Problems, Journal of Intelligent Systems and Internet of Things,Volume 1 , Issue 1 , PP: 48-60, 2020

**[25]** Mahmoud H.Alnamoly, Ahmed M. Alzohairy, Ibrahim M. El-Henawy, "A survey on gel images analysis software tools, Journal of Intelligent Systems and Internet of Things,Volume 1 , Issue 1 , PP: 40-47, 2021.

**[26]** S. Hemamalini ,V. D. Ambeth Kumar ,R. Venkatesan,S. Malathi. (2023). Relevance Mapping based CNN model with OSR-FCA Technique for Multi-label DR Classification. Journal of Fusion: Practice and Applications, 11 ( 2 ), 90-110.

**[27]** C. S. Manigandaa,V. D. Ambeth Kumar,G. Ragunath,R. Venkatesan,N. Senthil Kumar. (2023). De-Noising and Segmentation of Medical Images using Neutrophilic Sets. Journal of Fusion: Practice and Applications, 11 ( 2 ), 111-123.



DOI: https://doi.org/10.54216/JCIM.150110
Received: February 02, 2024 Revised: April 24, 2024 Accepted: July 26, 2024