

Multi-objective optimized task scheduling in cognitive internet of vehicles: towards energy-efficiency

M. Divyashree^{1,2}, H. G. Rangaraju³, C. R. Revanna¹

¹Department of Electronics and Communication Engineering, Government Sri Krishnarajendra Silver Jubilee Technological Institute, Bengaluru, Affiliated to Visvesvaraya Technological University, Belgaum, India

²Department of Electronics and Communication Engineering, RV Institute of Technology and Management, Bengaluru, India

³Department of Electronics and Communication Engineering, Government Engineering College, KR Pete, India

Article Info

Article history:

Received Apr 5, 2024

Revised Aug 29, 2024

Accepted Oct 1, 2024

Keywords:

Edge computing devices
Mobile edge computing
Multi-objective optimization
Multi-objective particle
swarm-optimization
Road side unit

ABSTRACT

The rise of intelligent and connected vehicles has led to new vehicular applications, but vehicle computing capabilities remain limited. Mobile edge computing (MEC) can mitigate this by offloading computation tasks to the network's edge. However, limited computational capacities in vehicles lead to increased latency and energy consumption. To address this, roadside units (RSUs) with cloud servers, known as edge computing devices (ECDs), can be expanded to provide energy-efficient scheduling for task computation. A new energy-efficient scheduling method called multi-objective optimization energy computation (MOEC) is proposed, based on multi-objective particle swarm optimization (MOPSO) to reduce ECDs' energy usage and execution time. Simulation results using MATLAB show that MOEC can balance the trade-off between energy usage and execution time, leading to more efficient offloading.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

M. Divyashree

Department of Electronics and Communication Engineering, Government Sri Krishnarajendra Silver Jubilee Technological Institute

K. R. Circle-560001, Bengaluru, Karnataka, India

Email: m.divyashree4@gmail.com

1. INTRODUCTION

With the proliferation of intelligent transportation systems, the volume of data produced by vehicles and their associated sensors has increased dramatically. Nevertheless, most vehicles lack the requisite capability for local data processing and storage. Consequently, computational responsibilities need to be transferred to distant cloud data centers. This is achieved through roadside units utilizing the vehicle-to-infrastructure (V2I) connection mode [1]. There are some drawbacks to this approach, in the internet of vehicles (IoV) network which experiences high transmission delays and inconsistent connections [2], [3].

Roadside units (RSUs) are strategically placed alongside road networks and highways to provide communication services for connected vehicles. To enhance the effectiveness of computational tasks for vehicles in cognitive internet of vehicles (CIoV), the function of RSUs has been extended to serve as edge computing devices (ECDs), offering processing and storage capabilities [4]. The mobile edge computing (MEC) [5]–[13] is an approach deploys cloud services closer to the radio access network's edge, facilitating the computation offloading [14] of tasks to nearby ECDs, located in close vicinity to the vehicles, instead of relying on distant cloud infrastructure [15]. In the framework of the CIoV, assigning tasks to ECDs can indeed enhance the quality of the driver's experience by addressing delays in transmission and improving connection stability [16]. However, when offloaded computational tasks accommodated in ECDs, it is crucial to prioritize the limited resources of these devices. Specifically, when operating on an ECD, there is a

necessity to restrict the quantity of concurrently active tasks. In such cases, there might be instances where computing tasks within the area of coverage of one ECD necessitate transfer to a different ECD for processing. Enhancing all ECDs' response times and lowering their energy usage are crucial for facilitating the computation of offloading between ECDs.

When a connected vehicle sends information to an ECD, the ECD calculates the energy required to process the information and also calculates the energy required by neighboring ECDs to process it. Based on this calculation, the ECD decides which neighboring ECD should process the information with less energy consumption. The information is then redirected to the selected ECD for processing [17]. This load-balancing technique [18], [19] helps with task scheduling [20], [21] by distributing tasks across the network, preventing any single ECD from becoming overloaded. By avoiding overloading, ECDs can achieve higher operational efficiency and consume lower amounts of energy. ECDs can function with increased efficiency, reduced energy consumption, and extended lifetime. This, in turn, helps to optimize the overall network's energy usage, making it more sustainable and cost-effective. The key contribution provided by this paper is implementation of multi-objective particle swarm optimization (MOPSO) to achieve multi-objective optimization, leading to decreased energy usage in ECDs and decrease execution time for computing tasks. This novel approach addresses the existing gaps in energy efficient computation for computational task allocation in CIOV, offering a more efficient and sustainable solution for intelligent transportation systems.

Numerous studies have explored energy-efficient strategies in edge computing/MEC implementation, with a focus on task scheduling/offloading in related papers. Ning *et al.* [17] proposed an MEC-enabled energy-efficient scheduling (MEES) method in IoV, which includes delay estimation, energy consumption estimation, task scheduling, processing, and result feeding back. The framework aims to minimize the energy consumption of RSUs while considering task latency constraints. They developed a heuristic algorithm that jointly considers task scheduling among MEC servers and downlink energy consumption of RSUs. The performance evaluations demonstrated the effectiveness of the framework in terms of energy consumption, latency, and task blocking possibility.

Liu *et al.* [22] introduced two computation offloading algorithms, binary offloading and partial offloading, in order to handle the issue of tasks being divided into indivisible and divisible tasks. The binary offloading method transfers the entire task to the MEC server and uses an enhanced method for upper confidence bounds to choose the best offloading site. The partial offloading algorithm divides complex tasks into time slots processed by different MEC servers, using the Q-learning algorithm to establish the most effective offloading strategy. The outcomes of the simulation imply that the binary offloading algorithm has lower delay cost and energy use during processing the computational intensive tasks, while the partial offloading algorithm significantly improves real-time performance and conserves mobile terminal energy.

Xu *et al.* [23] proposed an edge computing enabled computation offloading technique called edge computing offloading (ECO). It reduces computing task energy usage and execution time while addressing privacy conflicts. First, to acquire the routing vehicles from the origin vehicle in which the computing task is located to the destination vehicle, vehicle-to-vehicle (V2V) communication-based routing for a vehicle is developed. Then, non-dominated sorting genetic algorithm II (NSGA-II) is utilized to achieve the multi-objective optimization. Subsequent experimental evaluations verify the efficiency and effectiveness of ECO.

Behbehani *et al.* [24] developed a mixed integer linear programming (MILP) model that optimizes distribution of processing demands that comprise vehicles, computing in the edge also in the cloud. The model intends to lessen power usage, and compared to conventional clouds, the findings show power savings over 70%–90% for low workloads. However, for medium and large demand sizes, the results indicate a limited amount of cloud use due to capacity limitations on the vehicular and edge nodes, leading to 20%–30% power savings.

The structure of this paper is organized as follows. Section 2 presents the methodology, providing the mathematical modeling for problem formulation, the scheduling computation for CIOV in edge computing based on multi-objective particle swarm optimization, and the simulation environment. Section 3 presents and discusses the results. Finally, section 4 presents conclusion and future scope.

2. METHOD

This section analyzes mathematical models for problem formulation, including offloading time and energy usage estimation. It discusses a scheduling computation strategy for CIOV in edge computing environments, based on MOPSO. The strategy includes V2V transmission techniques for efficient offload path acquisition and computation scheduling, validated through a simulation setup.

2.1. Formulation of problems and the system model

This subsection presents a model for CIoV in cloud-edge computing systems, addressing a multi-objective optimization issue of computation scheduling. In cloud-edge computing, Figure 1 depicts a communication structure for CIoV [25]. Table 1 contains essential terms and their corresponding descriptions.

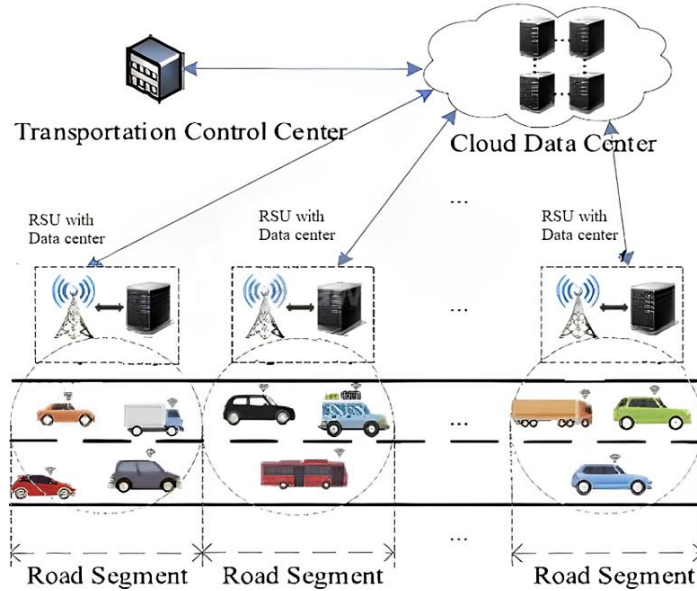


Figure 1. Communication structure for CIoV

Table 1. Defines the following key terms

Key term	Definition
M	Number of ECDs
D	Collection of ECD, where $D = \{d_1, d_2, \dots, d_m\}$
R	Collection of RSU, where $R = \{r_1, r_2, \dots, r_m\}$
S	Collection of servers, where $S = \{s_1, s_2, \dots, s_m\}$
N	The total count of vehicles
V	Collection of vehicles, where $V = \{v_1, v_2, \dots, v_n\}$
q	All-server capacity
T	Computing task, where $T = \{t_1, t_2, \dots, t_n\}$
t_n	The n^{th} computational task in T
u_n	Requested quantity of the t_n resource units
$Time_{total}$	Time consumed to implement T
E_{BL}	Baseline energy usage for all the servers
E_{ER}	Energy usage by resource units employed
E_{UR}	Energy usage by unemployeed resource units
E	Total energy usage by all the servers

2.1.1. Model of execution time

It is crucial to consider execution time, feedback period for sending back the execution's findings back to vehicle, and vehicle to ECD offloading time while using a vehicle to perform computations [23]. As the vehicles move alongside the road, they traverse multiple ECDs based on their current location. To determine whether a particular vehicle v_n , where $(n = \{1, 2, \dots, N\})$ is a part of the service sector of the m^{th} ECD at a given instant in time i , a flag is utilized. The flag is measured using:

$$F_n^m(i) = \begin{cases} 0, & v_n \text{ is within the coverage of } d_m \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

To transmit the t_n computational assignment to the intended destination segment's vehicle, V2V technology should be utilized. The time required for transmission of the t_n computing task, can be calculated using:

$$t_{TX}(i) = \sum_{m=1}^M \sum_{n'=1}^N F_n^m(i) \cdot Q_n^{n'}(i) \cdot \left(1 - F_n^m(i)\right) \cdot \frac{w_n}{\lambda_{v2v}} \cdot (\theta_{n,n'} + 1) \quad (2)$$

where $\theta_{n,n'}$ in the equation is the count of vehicles that were routed to $v_{n'}$ from v_n rate of data transmission using V2V technology is represented by λ_{V2V} , while binary variable $Q_n^{n'}(i)$ is used to determine, if t_n is delivered from v_n to $v_{n'}$ at a given time instant i . The calculation of $Q_n^{n'}(i)$ is given by:

$$Q_n^{n'}(i) = \begin{cases} 0, & t_n \text{ is transmitted from } v_n \text{ to } v_{n'} \\ 1, & \text{otherwise} \end{cases} \quad (3)$$

For the m^{th} ($m = \{1, 2, \dots, M\}$) computing task t_m , the duration of offloading is established by:

$$t_{OL}(i) = \sum_{m=1}^M F_n^m(i) \cdot \frac{w_n}{\lambda_{v2I}} \quad (4)$$

The rate of data transmission in V2I technology is denoted by λ_{V2I} . The duration required for task execution relies on both the task length and resource units' performance. If q represents the all-server capacity and u_n represents resource units quantity requested for t_n . The amount of time needed for the execution of the task t_n is:

$$t_{EX}(i) = \sum_{m=1}^M F_n^m(i) \cdot \frac{l_n}{u_n \cdot p} \quad (5)$$

Every resource unit possesses a processing power denoted by p . After the execution of a task, the vehicles must receive feedback regarding the results. The time required for this feedback can be calculated by (6):

$$t_{FB}(i) = \frac{w_n'}{\lambda_{v2I}} \quad (6)$$

Data's size in the output generated from the execution of t_n is denoted by w_n' . For the implementation of t_n , the overall time required is:

$$time_{total \text{ for } t_n}(i) = t_{TX}(i) + t_{OL}(i) + t_{EX}(i) + t_{FB}(i) \quad (7)$$

Subsequently, the overall duration required to implement all the tasks involving computation [23] is:

$$Time_{total} = \sum_{n=1}^N time_{total \text{ for } t_n}(i) \quad (8)$$

2.1.2. Model for energy usage

The amounts of energy used by ECDs are primarily attributed to the RSUs and the servers. As the amount of energy used by RSUs is dynamically regulated based on their employment status while they remain in operational mode, our attention is mainly on the servers. Amount of energy used by servers includes baseline energy consumed while they are running, energy used by the unoccupied resource units also energy utilized by resource units that are occupied [23]. The servers' service time is the primary factor in determining energy usage. The s_m service time can be computed by (9):

$$st_m(i) = \max_{n=1 \text{ to } N} (L_n^m(i) \cdot t_{EX}(i)) \quad (9)$$

The binary variable $L_n^m(i)$ assesses if t_n is carried out on s_m .

$$L_n^m(i) = \begin{cases} 0, & t_n \text{ is executed on } s_m \\ 1, & \text{otherwise} \end{cases} \quad (10)$$

Baseline energy consumption of all ECDs' servers' is:

$$E_{BL} = \sum_{m=1}^M st_m(i) \cdot \alpha \quad (11)$$

The ECD servers have a power rate denoted by the variable α . The process of determining the energy usage for resource units utilized employed is:

$$E_{ER} = \sum_{m=1}^M \sum_{n=1}^N L_n^m(i) \cdot st_m(i) \cdot \beta \quad (12)$$

The variable β represents the power rate of resource units that are employed. The process of determining the energy usage for resource units that are unemployed is carried out by:

$$E_{UR} = \sum_{m=1}^M (q - \sum_{n=1}^N L_n^m(i)) \cdot st_m(i) \cdot \gamma \quad (13)$$

The variable γ represents the power rate of resource units that are unemployed. Following that, the aggregate of all servers' energy usage [23] is determined using:

$$E = E_{BL} + E_{ER} + E_{UR} \quad (14)$$

2.2. Scheduling computation based on the multi-objective particle swarm optimization

Under this subsection, Firstly, the offloading path for the computational tasks is acquired through the adoption of V2V transmission. Furthermore, we utilize multi-objective optimization in order to determine the most effective scheduling strategy for these tasks, balancing objectives like minimizing execution time and reducing energy consumption. This approach maximizes overall performance in CIOV.

2.2.1. V2V transmission to offload path acquisition

In order to offload computing tasks to the goal ECD using V2V communication, a vehicle path to the final vehicle from the initial vehicle must be designed [26]. The offloading process can be optimized through strategic path design. This maximizes the potential of V2V communication in CIOV, reducing latency and enhancing computational efficiency.

Algorithm 1. Offloading path acquisition

Input: Vehicles (V), ECD

Output: Path (P)

Step 1: While vehicles are covered under the initial ECD do

Step 2: Calculate distance between two vehicles

Step 3: Calculate distance between the goal vehicles and the vehicles

Step 4: End while

Step 5: Choose the minimum distance between initial vehicle and the goal vehicle

Step 6: Return P

2.2.2. Computation scheduling based on multi-objective optimization energy computation (MOEC)

This subsection proposes a scheduling method for cloud-edge computing, addressing the multi-objective optimization problem with multiple goals. To address this, we employ MOPSO [27]–[31] which is an accurate and robust strategy in handling complex optimization tasks. MOPSO effectively balances multiple conflicting objectives, making it ideal for optimizing cloud-edge computing schedules. Here, we adopt MOEC to solve the energy-related aspects of the optimization problem. By integrating MOPSO, the proposed scheduling method MOEC ensures efficient and balanced task distribution, considering factors such as execution time and energy consumption.

- a. Initialize parameters: Define the swarm or population size, maximum iterations and other parameters like the inertial, cognitive, social coefficients and mutation rate, *etc.*
- b. Initialize population and repository: Population represents the collection of particles with their positions and velocities in search space. It determines each particle's fitness value in the initial population. Repository maintains non-dominated solutions obtained in the process of optimization. Initialize a population swarm of particles, each of which represents a solution, encoding the distribution of computing tasks T to the servers S . For each particle:
 - Randomly assign computing tasks to servers, respecting the all-server capacity q .
 - Initialize velocity and position vectors to guide search in solution space.
 Fitness evaluation: for each particle, calculate the objectives:
 - $Time_{total}$: total time usage for all tasks assigned.
 - E : compute total energy consumption using E_{BL} , E_{ER} , and E_{UR} considering the distribution of tasks and resource utilization.

Create a repository that stores the best non dominated solutions among the swarm to form a “Pareto front” that found so far. Non-dominated means no other solution is better in both objectives $Time_{total}$ and E .

- c. While (stop condition==false): Continue until a termination condition is met (Like getting a good enough solution quality or reaching a maximum count of iterations).
- d. Choose leader “ h ”: Choose a leader from the population or repository to direct the motion of each particle. To maintain a diverse collection of solutions, the leader should be chosen according to its Pareto optimality and typically a diversity mechanism.

- e. Update positions and velocities: Update velocity for each particle according to its current velocity, distance to that's personal best position and distance to the chosen leader 'h'. Each particle's position should be updated according to the new velocity. The new position represents its new solution.
- f. Perform mutation (if necessary): Apply mutation by introducing random perturbations or alterations to some particles to explore new regions in search space if desired to add diversity to the population and prevent too rapid convergence.
- g. Boundary checks on position and velocity: Check that the new positions and velocities stay within the defined boundaries. Bring any element of a position or velocity vector back inside the boundaries if it goes beyond.
- h. Update the optimal positions: Update personal best positions and global best positions based on dominance or fitness criteria. Use objectives ' $Time_{total}$ ' and ' E ' to evaluate the fitness of new positions. Update a particle's personal best in terms of Pareto dominance if, its new position is better than its previous one. Add new non-dominated solutions to the repository, ensuring that it eliminates duplicates and maintains its representation of the current.
- i. Termination check: The algorithm continues iterating until set counts of iterations have passed, no significant improvement is observed, or another stopping condition is satisfied.
- j. Result: Output the Pareto front that represents the best trade-offs found between time consumption $Time_{total}$ and total energy consumption E for executing computing tasks on servers.

As a result, the MOPSO algorithm updates particle velocities, positions and the repository to find non-dominated solutions for multi-objective optimization problems. The iterative process continues, with particles exploring and exploiting the search space, until termination conditions are met. Upon meeting these conditions, the algorithm concludes by identifying optimal or near-optimal solutions that balance the multiple objectives of the optimization problem.

2.3. Simulation setup

This subsection includes several extensive simulations carried out to evaluate effectiveness of suggested scheduling method for edge computing, referred as MOEC by using MATLAB 2020 which introduces the MATLAB 9.8 runtime. It enables parallel computing for hundreds of functions. This feature allows users to use local multicore processors and graphics processing units (GPUs), and scale computations to compute clusters, improving performance and productivity in large-scale data processing tasks.

For our simulation, we consider varying numbers of vehicles, specifically 20, 40, 60, 80, 100, and 120. The rate of data transmission using V2V technology denoted as λ_{V2V} , and the rate of data transmission using V2I technology, denoted as λ_{V2I} , are both set to 1 Gbps and 600 Mbps respectively, following the values mentioned in [1], [12]. The simulation parameter settings used are represented in Table 2. The Simulation setup was used to test the system models for problem formulation and scheduling computation strategies. Subsequently, the proposed MOEC performance is assessed on vehicle scales in terms of time and energy usage.

Table 2. Simulation parameter settings

Description of the parameter	Value
Data transmission rate using V2V technology λ_{V2V}	1 Gbps
The entire number of ECDs is M	20
The ECDs servers have a power rate of α	300 W
Employed resource units have a power rate of β	50 W
Unemployed resource units have a power rate γ	30 W
Data transmission rate using V2I technology λ_{V2I}	600 Mbps
Every resource unit possesses a processing power	2000 MHz

3. RESULTS AND DISCUSSION

The proposed scheduling or offloading MOEC method is compared to the existing ECO method for a comprehensive comparative analysis. To illustrate the variations and efficacy of these methods, a thorough comparison is provided. This paper employs a comparative approach, which is detailed in the following subsections.

3.1. Existing ECO

The previously developed ECO [15] method's purpose is to strike a balance between optimizing the use of time and lowering energy consumption. The method involves conducting multiple experiments to evaluate its performance and identify optimal solutions. To determine a set of comparatively superior solutions, multiple-criteria decision-making (MCDM) with simple additive weighting (SAW) techniques

were employed. These techniques help in assessing and selecting the best offloading strategies based on the fitness of the solutions regarding time and energy objectives.

3.2. Analysis of comparison

This paper provides a detailed comparison of the proposed MOEC methodology with the ECO method under same experimental conditions. Execution time and energy usage are the two primary parameters used to evaluate the performance. To illustrate the real resource utilization of all ECDs participating in hosting the computing tasks, number of employed ECDs, resource utilization, and number of computational tasks offloaded across the ECDs show the outcomes.

3.2.1. The usage of ECDs- a comparison

Both ECO and proposed MOEC approaches are compared, along with the amount of ECDs used. According to Figure 2, a total of 20 ECDs are used in this experiment. The MOEC technique employs ECDs more frequently as the number of vehicles increases, as illustrated in the figure. Notably, All ECDs need to be operational in order to meet the requirements for the deployment of the computing tasks once there are 100 vehicles.

MOEC is highly scalable and efficient in handling higher loads. While ECO employs fewer ECDs across all vehicle numbers, indicating potential limitations in handling larger numbers. MOEC's significant increase in ECD usage as vehicle numbers rise supports its potential for real-world applications where demand can vary greatly.

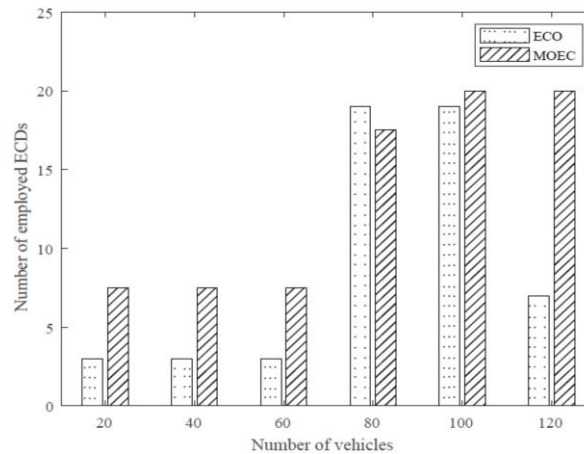


Figure 2. The quantity of ECDs employed at various vehicle size by ECO and proposed MOEC: a comparison

3.2.2. The usage of resources- a comparison

It is guaranteed that the resource unit will be occupied once all computing jobs have been transferred to the ECDs using the appropriate techniques. Figure 3 compares the resource utilization of the ECDs by the ECO and the proposed MOEC at different vehicle scales. The resource utilization is determined by estimating the number of engaged ECDs and the amount of resource units allotted to each ECD. Higher resource utilization results from allocating more resource units but using fewer ECDs. The findings illustrated in Figure 3 demonstrate that MOEC consistently outperforms ECO in resource utilization across various vehicle scales, with an estimated 80% utilization rate.

Efficient resource utilization is essential for addressing increased demand and reducing waste. The higher resource utilization rates of MOEC, compared to ECO, validate that MOEC would be more effective in resource management. MOEC's consistent 80% utilization indicates its capability to handle various operational demands efficiently.

3.2.3. Amount of computational task offloaded among ECDs: a comparison

In common practice, the computing task is delegated to the neighboring ECD. However, in cases when the vehicle sizes are modest, there is a random distribution of the vehicles among different ECD ranges. Under these conditions, assigning all computational tasks to the surrounding ECDs may lead to many ECDs being active simultaneously, resulting in excessive energy consumption. We solve this in our

experiment by offloading the computational burden from the surrounding ECD to a neighboring ECD. When the area of coverage of the destination ECD differs from that of the origin ECD, this allows the offloading of computing task across ECDs. Figure 4 compares both ECO and proposed MOEC techniques for the amount of computational tasks offloaded among ECDs. It shows that MOEC efficiently utilizes resources by distributing computational tasks across ECDs as vehicle size increases. Despite both ECO and MOEC showing an increase in offloaded tasks, ECO offloads more tasks for most vehicle counts, especially when vehicles exceed 60.

ECO's higher offloaded tasks indicate reliance on external devices, causing higher energy and time consumption. MOEC's fewer offloaded tasks suggest local computations or efficient task management strategies, reducing offloading. As the system scales, both algorithms' offloaded tasks reflect growing computational demand. MOEC's superior efficiency in managing resources ensures better performance as the number of vehicles increases.

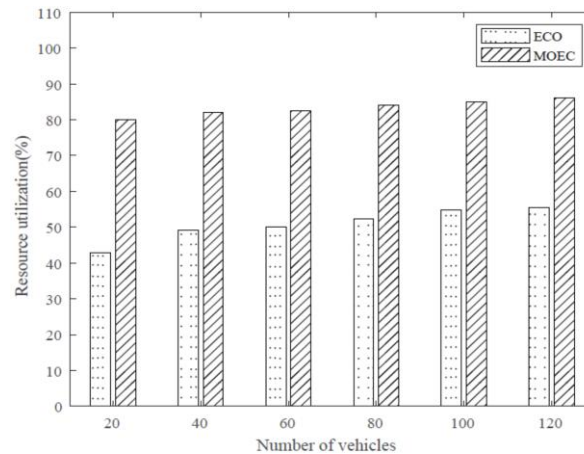


Figure 3. Resource usage between ECO and proposed MOEC at various vehicle size: a comparison

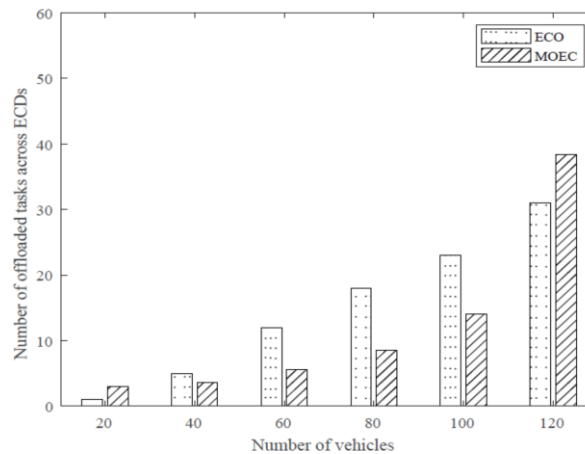


Figure 4. Amount of computational task offloaded among ECDs by ECO and proposed MOEC at various vehicle size: a comparison

3.2.4. Energy usage: a comparison

As stated in Section 3, there are three parts to energy consumption: the baseline energy use for all the servers in the ECDs, the energy usage of resource units which are in use, and the energy usage of the units that are not in use. The results reveal these three energy usage factors for ECO and the proposed MOEC techniques across various vehicle scales in Figure 5. It is evident from Figure 5(a) that when the vehicle scale grows, ECO shows an increase in baseline energy consumption for all the servers in the ECDs especially beyond 60 vehicles, reaching up to about 2 kWh at 120 vehicles. However, the MOEC approach consumes

less energy than ECO since it employs fewer ECDs. The energy usage of employed resource units across the various vehicles size is seen in Figure 5(b). As it uses almost the same amount of resource units for computing activities, the MOEC technology achieves comparable energy consumption of the utilized resource units across various vehicle scales. In Figure 5(c) the MOEC strategy yields slightly higher energy consumption from idle resource units which can be viewed in the context of its overall optimization goals. Energy usage comparison in Figure 6, underlines the MOEC method's superior performance even further. As an instance, if there are over 100 vehicles, less than 1 kWh of energy is used by the MOEC method, whereas ECO uses more energy than 1 kWh.

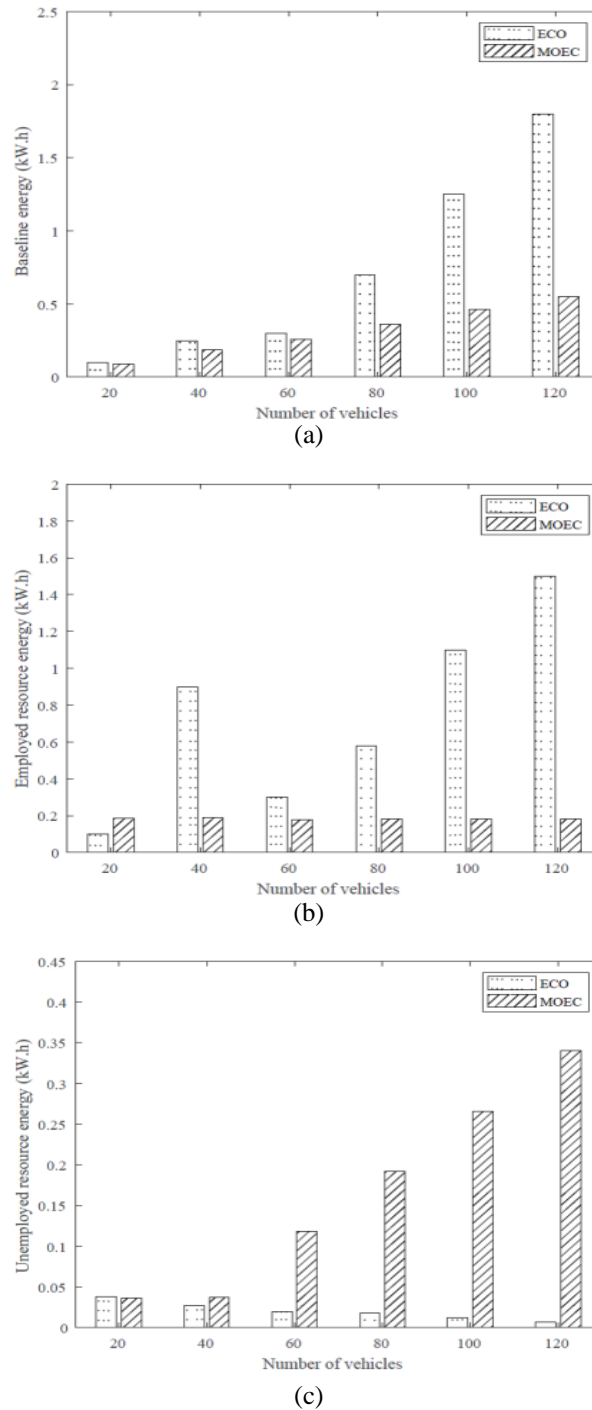


Figure 5. Proposed MOEC and ECOs comparison of several energy consumption factors at various vehicle size (a) baseline energy usage by all the servers in the ECDs', (b) energy usage by resource units employed, and (c) energy usage by idle resource units

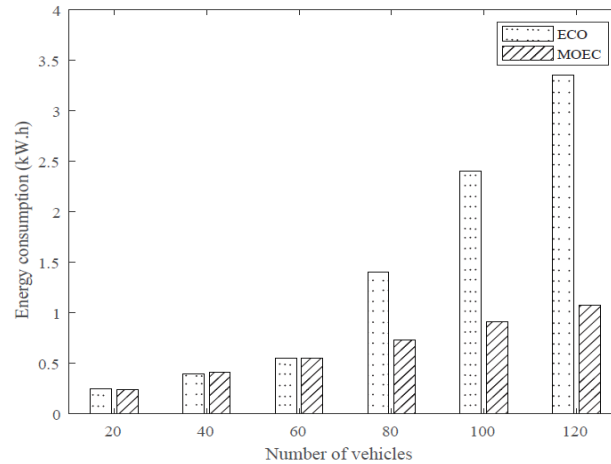


Figure 6. Energy usage by proposed MOEC and the ECO at various vehicle size: a comparison

These findings reveal that MOEC is more energy-efficient and scalable than ECO, highlighting the significance of energy efficiency in edge computing. MOEC's lower baseline, employed, and unemployed resource energy requirements make it suitable for large-scale implementations. While ECO's higher reliance on offloading tasks results in increased energy usage.

3.2.5. Time consumed: a comparison

The offloading time and total time consumption between the proposed MOEC and the ECO method are compared at various vehicle scales in Figures 7 and 8 respectively. An essential statistic for calculating time consumption is the offloading time. Figure 7 clearly shows that MOEC method has quicker offloading times than the ECO method across all vehicle scales. The average time for offloading tasks in ECO is 0.5 seconds for 20 vehicles, while MOEC takes 0.6 seconds. As the number of vehicles increases, the gap widens, with MOEC taking about 1 second for 120 vehicles, highlighting its superior efficiency. In Figure 8, we compared the total time used about the two offloading techniques. The ECO method requires time longer than the MOEC solution especially beyond 80 vehicles, indicating that ECO requires more time to complete tasks as vehicle numbers grow.

The MOEC method outperforms the ECO method in offloading time and total time consumption, especially as vehicle count increases. MOEC's efficient scheduling and task management strategies result in lower offloading times, maintaining around 1 second even with 120 vehicles, compared to ECO's 4.5 seconds. MOEC also shows lower total time consumption, indicating superior scalability. This highlights MOEC's potential for practical traffic management applications, maintaining efficiency and minimizing delays under high computational demands.

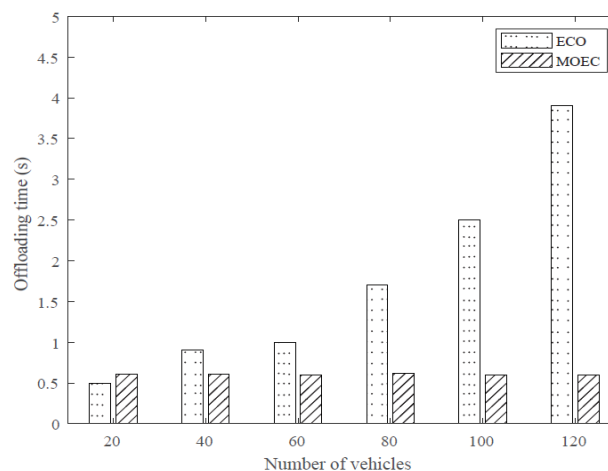


Figure 7. Offloading time usage of ECO and proposed MOEC at various vehicle size: a comparison

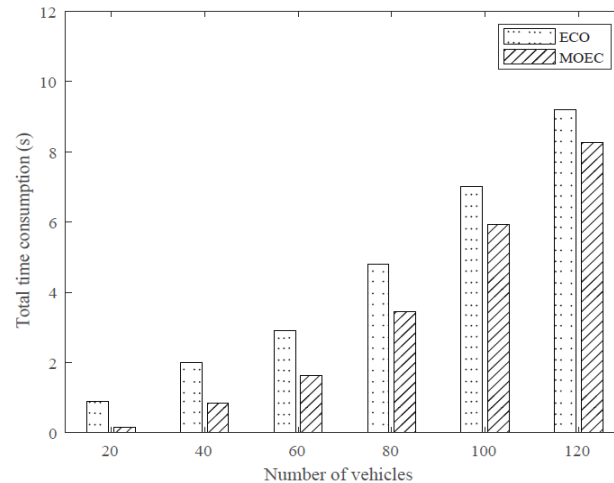


Figure 8. Time usage of ECO and proposed MOEC at various vehicle size: a comparison

4. CONCLUSION

The rapid advancement of CIoV technology has led to complex computational demands, necessitating the offloading of tasks to remote infrastructure. Among the various paradigms available for handling these tasks, the MEC framework has proven to be highly effective. It involves offloading the vehicle's computational tasks to the ECDs located in close proximity. This study uses MOEC, a computation scheduling technique leveraging edge computing, to achieve multi-objective optimization, decreasing both execution time of computational tasks and energy usage of ECDs in CIoV environments.

In the proposed MOEC approach initially, origin vehicle establishes V2V communication-based routing to destination vehicle by recognizing routing vehicles. MOPSO is then used to accomplish multi-objective optimization to enhance the efficiency of task scheduling within the MOEC framework. Later experimental assessments confirm the effectiveness and efficiency of MOEC.

The future goal is to further predict traffic congestion in CIoV systems by the use of advanced algorithms for machine learning and artificial intelligence techniques. This could use predictive modeling to better anticipate traffic patterns which enhances efficiency, reduces travel time, and improves user experience. This contributes to the advancement of more responsive transportation systems.

This research addresses challenges in CIoV and lays the groundwork for future intelligent transportation systems by incorporating edge computing and predictive analytics. It proposes innovative solutions to enhance efficiency, safety, and reliability of transportation networks. It aims to optimize data processing and decision-making processes, collaborating to the evolution of smarter and more adaptive transportation systems.




REFERENCES

- [1] L. Kong, M. K. Khan, F. Wu, G. Chen, and P. Zeng, "Millimeter-wave wireless communications for IoT-cloud supported autonomous vehicles: overview, design, and challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 62–68, Jan. 2017, doi: 10.1109/MCOM.2017.1600422CM.
- [2] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 12, pp. 10660–10675, Dec. 2017, doi: 10.1109/TVT.2017.2714704.
- [3] R. Deng and H. Liang, "Whether to charge or discharge an electric vehicle? An optimal approach in polynomial time," in *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, Sep. 2017, pp. 1–5, doi: 10.1109/VTCFall.2017.8288324.
- [4] M. Wang, J. Wu, G. Li, J. Li, Q. Li, and S. Wang, "Toward mobility support for information-centric IoV in smart city using fog computing," in *2017 IEEE International Conference on Smart Energy Grid Engineering (SEGE)*, Aug. 2017, pp. 357–361, doi: 10.1109/SEGE.2017.8052825.
- [5] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.
- [6] Q.-V. Pham *et al.*, "A survey of multi-access edge computing in 5G and beyond: fundamentals, technology integration, and state-of-the-art," *IEEE Access*, vol. 8, pp. 116974–117017, 2020, doi: 10.1109/ACCESS.2020.3001277.
- [7] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017, doi: 10.1109/ACCESS.2017.2685434.
- [8] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: partial computation offloading using dynamic voltage scaling," *IEEE Transactions on Communications*, vol. 64, no. 10, pp. 4268–4282, 2016, doi: 10.1109/TCOMM.2016.2599530.
- [9] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018, doi: 10.1109/TVT.2018.2876804.
- [10] J. Du, L. Zhao, X. Chu, F. R. Yu, J. Feng, and C.-L. I, "Enabling low-latency applications in LTE-A based mixed fog/cloud




- computing systems,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1757–1771, 2019, doi: 10.1109/TVT.2018.2882991.
- [11] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, “A survey on mobile edge computing: The communication perspective,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017, doi: 10.1109/COMST.2017.2745201.
- [12] M. B. M. Mansour, T. Abdelkader, M. H. AbdelAziz, and E.-S. M. El-Horbaty, “A trust evaluation scheme of service providers in mobile edge computing,” *International Journal of Electrical and Computer Engineering*, vol. 12, no. 2, pp. 2121–2138, Apr. 2022, doi: 10.11591/ijece.v12i2.pp2121-2138.
- [13] M. El Ghmary, Y. Hmimz, T. Chanyour, and M. O. Cherkaoui Malki, “Time and resource constrained offloading with multi-task in a mobile edge computing node,” *International Journal of Electrical and Computer Engineering*, vol. 10, no. 4, pp. 3757–3766, Aug. 2020, doi: 10.11591/ijece.v10i4.pp3757-3766.
- [14] S. Maftah, M. El Ghmary, H. El Bouabidi, M. Amnai, and A. Ouacha, “Intelligent task processing using mobile edge computing: processing time optimization,” *IAES International Journal of Artificial Intelligence*, vol. 13, no. 1, pp. 143–152, Mar. 2024, doi: 10.11591/ijai.v13.i1.pp143-152.
- [15] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. ZHANG, “Mobile-edge computing for vehicular networks: a promising network paradigm with predictive off-loading,” *IEEE Vehicular Technology Magazine*, vol. 12, no. 2, pp. 36–44, Jun. 2017, doi: 10.1109/MVT.2017.2668838.
- [16] M. Chen, Y. Tian, G. Fortino, J. Zhang, and I. Humar, “Cognitive Internet of vehicles,” *Computer Communications*, vol. 120, pp. 58–70, May 2018, doi: 10.1016/j.comcom.2018.02.006.
- [17] Z. Ning, J. Huang, X. Wang, J. J. P. C. Rodrigues, and L. Guo, “Mobile edge computing-enabled Internet of vehicles: toward energy-efficient scheduling,” *IEEE Network*, vol. 33, no. 5, pp. 198–205, Sep. 2019, doi: 10.1109/MNET.2019.1800309.
- [18] X. He, Z. Ren, C. Shi, and J. Fang, “A novel load balancing strategy of software-defined cloud/fog networking in the internet of vehicles,” *China Communications*, vol. 13, no. Supplement2, pp. 140–149, 2016, doi: 10.1109/CC.2016.7833468.
- [19] S. K. Maurya, S. Malik, and N. Kumar, “Virtual machine tree task scheduling for load balancing in cloud computing,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 388–393, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp388-393.
- [20] A. H. Shamman, H. A. Alasadi, H. A. Ameen, Z. I. Rasol, and H. M. Gheni, “Cost-effective resource and task scheduling in fog nodes,” *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 27, no. 1, pp. 466–477, Jul. 2022, doi: 10.11591/ijeecs.v27.i1.pp466-477.
- [21] N. K. Chowdaiah and A. Dammur, “Resource-efficient workload task scheduling for cloud-assisted internet of things environment,” *International Journal of Electrical and Computer Engineering*, vol. 13, no. 5, pp. 5898–5907, Oct. 2023, doi: 10.11591/ijece.v13i5.pp5898-5907.
- [22] J. Liu, S. Wang, J. Wang, C. Liu, and Y. Yan, “A task oriented computation offloading algorithm for intelligent vehicle network with mobile edge computing,” *IEEE Access*, vol. 7, pp. 180491–180502, 2019, doi: 10.1109/ACCESS.2019.2958883.
- [23] X. Xu *et al.*, “An edge computing-enabled computation offloading method with privacy preservation for Internet of connected vehicles,” *Future Generation Computer Systems*, vol. 96, pp. 89–100, Jul. 2019, doi: 10.1016/j.future.2019.01.012.
- [24] F. S. Behbehani, M. Musa, T. Elgorashi, and J. M. H. Elmirghani, “Energy-efficient distributed processing in vehicular cloud architecture,” in *2019 21st International Conference on Transparent Optical Networks (ICTON)*, Jul. 2019, pp. 1–4, doi: 10.1109/ICTON.2019.8840335.
- [25] M. Divyashree, H. G. Rangaraju, and C. R. Revanna, “Mobile adaptive routing algorithm for road-aware infrastructure-assisted communication in cognitive Internet of vehicles,” *International Journal of Interactive Mobile Technologies (IJIM)*, vol. 18, no. 4, pp. 97–111, Feb. 2024, doi: 10.3991/ijim.v18i04.44715.
- [26] X. Xu, R. Gu, F. Dai, L. Qi, and S. Wan, “Multi-objective computation offloading for internet of vehicles in cloud-edge computing,” *Wireless Networks*, vol. 26, no. 3, pp. 1611–1629, Apr. 2020, doi: 10.1007/s11276-019-02127-y.
- [27] S. Lalwani, S. Singhal, R. Kumar, and N. Gupta, “A comprehensive survey: Applications of multi-objective particle swarm optimization (MOPSO) algorithm,” *Transactions on Combinatorics*, vol. 2, no. 1, pp. 39–101, 2013.
- [28] P. Liu, Y. Fan, X. Xiong, Y. Wen, and D. Lu, “MOPSO-based data scheduling scheme for P2P streaming systems,” *KSII Transactions on Internet and Information Systems*, vol. 13, no. 10, pp. 5013–5034, Oct. 2019, doi: 10.3837/tiis.2019.10.011.
- [29] Q. Zhang, Y. Liu, H. Han, M. Yang, and X. Shu, “Multi-objective particle swarm optimization with multi-archiving strategy,” *Scientific Programming*, vol. 2022, no. 1, pp. 1–21, May 2022, doi: 10.1155/2022/7372450.
- [30] F. Ramezani, J. Lu, and F. Hussain, “Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization,” in *Proceedings of the 2013 IEEE Congress on Evolutionary Computation (CEC)*, 2013, pp. 237–251, doi: 10.1007/978-3-642-45005-1_17.
- [31] M. Zhang, L. Liu, C. Li, H. Wang, and M. Li, “A particle swarm optimization method for AI stream scheduling in edge environments,” *Symmetry*, vol. 14, no. 12, pp. 1–18, Dec. 2022, doi: 10.3390/sym14122565.

BIOGRAPHIES OF AUTHORS






M. Divyashree    is a research scholar in the Department of Electronics and Communication Engineering at Government SKSJT Institute, Bengaluru, Karnataka, India, and is currently working as an assistant professor in the Department of Electronics and Communication Engineering at RV Institute of Technology and Management, Bengaluru, Karnataka. She completed her B.E. in telecommunication engineering from GSSSIETW, Mysuru, and M.Tech. in digital electronics and communication systems from VTU-Regional Center, Mysuru. She has about 7 years of teaching experience and has published 08 papers in international and national journals and conferences. Her areas of interest include wireless communication, wireless ad-hoc networks, and the internet of things. She can be contacted at email: m.divyashree4@gmail.com.



H. G. Rangaraju    is currently working as associate professor in the Electronics and Communication Engineering Department at Government Engineering College, K R Pet, Karnataka, India. He received B.E. degree in electronics and communication engineering from SIT, Tumkur, M.E. degree in electronics and communication engineering from University Visvesvaraya College of Engineering, Bengaluru University, and Ph.D. degree from Visvesvaraya Technological University, Belagavi. He has more than 20 years of experience in teaching and industry. He has to his credit two patents and more than 25 research publications in national/international journals and conferences. Now, he is guiding four research scholars for their PhD degrees under VTU. His major areas of interest include VLSI design, signal processing, communications, and wireless networks. He can be contacted at email: rangraju@gmail.com.



C. R. Revanna    from Chikmagalur District of Karnataka State, India. He is currently serving as assistant professor and head of the Department of Electronics and Communication Engineering at Government SKSJ Technological Institute, Bengaluru. He has 25 years of teaching experience in various institutions under the Department of Technical Education, GoK. He obtained his PhD degree for his study on cryptography from Jain University, Bangalore. He has been engaged in research for over a decade and has published many articles in renowned international journals in different domains. He is a member of the Board of Examiners for different universities and academic institutions. He is a reviewer for many international journals and has chaired international conferences. He is actively involved in the formulation of HEI-enabled Institutional National Innovation and Startup Policy. He can be contacted at email: revannacr2008@gmail.com.