Online-Journals.org

iTDAF IETI Transactions on
Data Analysis and Forecasting

iTDAF | eISSN: 2959-0442 | Vol. 2 No. 2 (2024) | OPEN ACCESS

https://doi.org/10.3991/itdaf.v2i2.50949

PAPER

Optimizing Energy and Delay in Task Offloading for Connected Vehicles with Proximal Policy Optimization

Salman Raza(✉)

Department of Computer Science, National Textile University Faisalabad, Faisalabad, Punjab, Pakistan

salmanraza@ntu.edu.pk

ABSTRACT

Electric-powered intelligent connected vehicles are becoming the pivotal point of the automotive industry. As vehicles integrate more applications, the computation tasks they generate increase substantially. Cloud servers cannot handle these tasks promptly, and current electric vehicles (EVs) have limited energy and computing resources. Multi-Access edge computing (MEC) performs various activities in proximity to the vehicles, resulting in decreased latency and the preservation of EV battery power. However, MEC servers have finite processing resources and may be unable to satisfy the required latency restrictions. We propose a task offloading scheme to optimize the allocation of computational resources from roadside servers across several EVs. We develop a mathematical model to optimize both computation latency and EV energy, represented as a Markov decision process (MDP). To address this, we employ the deep reinforcement learning-proximal policy optimization (DRL-PPO) algorithm. The implementation of our mathematical model, which is based on an MDP, together with the use of the DRL-PPO algorithm, showcases notable decreases in both energy consumption and latency when compared to alternative benchmark deep reinforcement learning (DRL) approaches.

KEYWORDS

multi-access edge computing (MEC), task offloading, deep reinforcement learning (DRL)

1 INTRODUCTION

The advent of autonomous driving (AD) and the proliferation of 5G networks have significantly increased interest in intelligent connected vehicles [1]. Simultaneously, the demand for low-carbon emissions and the rising cost of fossil fuels are driving a shift towards electric vehicles (EVs) [2]. Despite the benefits, sophisticated AD technology generates substantial computation tasks, and passengers increasingly engage in recreational activities, further escalating energy consumption. Current intelligently connected EVs face challenges due to limited battery capacity [3]. Additionally, the concurrent generation of multiple computation tasks for AD, entertainment, and work can exceed the vehicles' computational capabilities.

Raza, S. (2024). Optimizing Energy and Delay in Task Offloading for Connected Vehicles with Proximal Policy Optimization. *IETI Transactions on Data Analysis and Forecasting (iTDAF)*, 2(2), pp. 29–40. https://doi.org/10.3991/itdaf.v2i2.50949

Article submitted 2024-04-06. Revision uploaded 2024-06-23. Final acceptance 2024-07-10.

To mitigate these issues, computation tasks can be offloaded to the cloud or multi-access edge computing (MEC) servers. However, cloud servers are often distant, resulting in long transmission latencies that do not meet stringent requirements. Deploying MEC servers at roadside units (RSUs) offer an efficient solution due to their superior computational capacity, reducing delay and conserving EV battery power, as transmitting tasks to MEC servers consumes less energy than local processing [4].

Table 1. Summary of important acronyms

Acronyms	Definition	Acronyms	Definition
MEC	Mobile Access Edge Computing	AC	Actor-Critic
EV	Electric Vehicles	DQN	Deep Q-Network
AD	Autonomous Driving	DRL-PPO	Deep Reinforcement Learning
RSU	Road Side Unit	PPO	Proximal Policy Optimization
MDP	Markov Decision Process	UAV	Unmanned Aerial Vehicle

Nevertheless, an unbalanced load on MEC servers arises if all EVs offload their tasks simultaneously, due to uneven temporal and spatial vehicle distributions [5]. Therefore, an effective offloading strategy and computing resource allocation are crucial. Different computation tasks have varying latency requirements; for instance, intelligent traffic scheduling demands high latency, navigation requires moderate latency, and video tasks, which are larger, can tolerate lower latency [6]. While local processing might consume more energy, it can be advantageous when the tasks fall within the vehicle's computational limits, as it eliminates communication latency.

Existing study primarily addresses three offloading strategies: full offloading, binary offloading, and partial offloading. Using edge servers for all designed tasks is known as full offloading, and it can drastically lower the energy usage of EVs. Nevertheless, this approach results in a higher processing latency at the edge servers, where numerous requests gather, although these servers lack adequate capacity to fulfill the service demand. Binary offloading, analogously to the previous case, means a set of certain operations that are wholly transferred to the edge server. This approach is suitable for small tasks, like IoT services, but falls short for larger tasks, as edge servers with constrained resources can support only a limited number of vehicles, providing marginal performance improvements. Processing the entire task locally, on the other hand, results in high latency and significant energy consumption. Partial offloading, advocated for, divides tasks into segments and processes some segments off with edge servers while tackling the remainder locally [7]. It is useful for improving the efficiency of resources and minimizing delays. Several works have been presented focusing on different aspects of partial offloading. For example, one study suggested a mobility-aware partial offloading strategy to lower the computational cost of the system by utilizing the vehicle's processing capability to lessen the burden on the MEC server [8]. Another study focuses on partial offloading, meaning that some tasks should be offloaded to RSUs while at the same time multiple access technologies are employed to optimize the combined energy and delay [9]. According to these studies, partial offloading positively impacts the utilization of computation resources in both vehicles and MEC servers, leading to a reduction in task latency.

[10] presents a distributed DRL-based technique called P-D3QN for optimum job offloading. This approach aims to reduce system latency while taking into account

the limitations of unmanned aerial vehicle (UAV) resources. The experimental findings indicate that the P-D3QN method enhances latency by 26.24% and augments offloading utility by 42.26% in comparison to benchmark schemes. The authors in [11] suggested method for optimizing hybrid energy storage system power management and setup is an integrated approach that combines the non-dominated sorting genetic algorithm III (NSGA-III) with fuzzy logic-based control. The utilization of this technology greatly enhances the lifespan of the battery and decreases the cost-to-range ratio of the energy storage system. The optimized hybrid energy storage system improves the battery's lifespan by more than 72% and reduces its weight by 34.3% compared to using a standalone battery energy storage system. The fuzzy-logic expectation-maximization technology surpasses conventional rule-based approaches and may be seamlessly incorporated into vehicles in real-time.

However, the majority of recent partial offloading study emphasizes delay rather than energy consumption, a critical limitation for EVs. This paper aims to minimize both delay and energy consumption while selecting an appropriate approach for partial offloading. It proposes an optimized partial offloading strategy that considers both aspects. The strategy distributes the computation load between the neighboring RSU and the RSUs in the vehicle's direction, thereby balancing the MEC servers' workload and enhancing overall system efficiency. We introduce a mobility-aware task offloading strategy, optimizing resource allocation from RSU servers for multiple EVs. A mathematical model based on a Markov decision process (MDP) is developed to minimize computing delay and energy consumption in EVs. This model is addressed using the deep reinforcement learning-proximal policy optimization (DRL-PPO) technique. Our simulations validate the performance improvements of the proposed approach.

The following are the key contributions to this paper:

- To address the issues of delay and energy consumption in task offloading for EVs by formulating an optimization model.
- To address uneven vehicle distribution, we intend to study resource allocation among RSUs to balance workloads and improve efficiency.
- To develop a mathematical model to optimize computation delay and EV energy as an MDP using the DRL-PPO algorithm.

The rest of this paper is structured as follows: Section 2 outlines the system model under consideration. In Section 3, we formulate the problem, and Section 4 introduces the proposed scheme. Section 5 provides the numerical results and discussions, while Section 6 concludes the paper.

2 SYSTEM MODEL

We consider an urban highway scenario with four lanes, where each RSU is made up of a MEC server and covers distinct, non-overlapping areas. By helping to process computing activities, these MEC servers lessen the load on EVs. With its communication and computing capabilities, every EV sends data, including variables like velocity, battery life, distance from the RSU, and accessible resources, to the RSU in real time. Divide tasks into many parts and offload some of them to the closest RSU to optimize resource utilization while minimizing delay and energy consumption.

The RSU searches for any other available MEC servers to take on the responsibilities along the vehicle's route. Tasks that can be offloaded to a single MEC server are generated by EV, and after all results are received, the task is considered effectively processed. Vehicles located in the MEC servers' service area are represented by $K = \{1, \ldots, k\}$, while the MEC servers that are deployed at RSUs are indicated by $R = \{1, \ldots, r\}$. We also establish a set of tasks for every vehicle in it. The vehicle's k computing task at time t is represented by the notation $I_{t,k}$.

2.1 System framework

We partition the task $I_{t,k}$ into two separate segments: one processed on-vehicle (locally) and the other offloaded to the MEC server. Let $0 \le p_{I_{t,k}} \le 1$ represent the fraction of $I_{t,k}$ handled locally, with $1 - p$ indicating the portion offloaded. A task is considered successfully completed only when both segments are fully processed, irrespective of whether it is executed on vehicle or partially offloaded. The method for determining the data transmission rate between a vehicle and an MEC server is based on the approach detailed in [8]. The bandwidth is represented as $B_{I_{tk}}$ between the MEC and vehicular task $I_{t,k}$. This allocation ensures that each task has a dedicated portion of the communication channel, facilitating efficient data transfer between the vehicle and the server. The transmission power of vehicle is P_k^r, while h_k^r represents the channel gain. Additionally, σ indicates the Gaussian noise power as described in [12]. As a result, the V2I transmission rate R_k^r can be written as follows:

$$R_k^r = B_k^r \log_2\left(1 + \frac{P_k^r \cdot h_k^r}{\sigma}\right) \tag{1}$$

Moreover, the task $I_{t,k}$ uploading latency is expressed by

$$T_{I_{t,k}}^{up} = \frac{\left(1 - p_{I_{t,k}}\right) C_{I_{t,k}}}{R_k^r} \tag{2}$$

Where $C_{I_{t,k}}$ denotes the task $I_{t,k}$ size.

During the task upload process, we focus solely on the energy consumption of the vehicles that is expressed as follows:

$$E_{I_{t,k}}^{V2I} = T_{I_{t,k}}^{up} P_k^r = \frac{\left(1 - p_{I_{t,k}}\right) C_{I_{t,k}} P_k^r}{R_k} \tag{3}$$

2.2 Local computing model

Given that the vehicle possesses its processing resources, a task can be computed locally, thereby saving the time required for uploading and downloading data. However, this process consumes the vehicle's own energy. The time consumed for locally executing the task $I_{t,k}$ can be described as follows:

$$T_{I_{t,k}}^{local} = \frac{p_{I_{t,k}} D_{I_{t,k}}}{f^k} \tag{4}$$

The energy consumption is expressed as:

$$E_{I_{t,k}}^{local} = K(f^k)^2 p_{I_{t,k}} D_{I_{t,k}} \tag{5}$$

where *D* {*I*_ {*t*, *k*}} represents the CPU cycles required to execute task $I_{t,k}$, and *f*^*kk* is the vehicle CPU frequency and *K* being the energy coefficient.

2.3 V2I computing model

To guarantee timely task completion, tasks are offloaded either to the closest or to the subsequent RSU along the vehicle's route. This decision is strategically made to ensure that computational resources are utilized optimally. Leveraging the computational resources of the MEC for task processing can significantly reduce both delay and energy consumption, making it a more efficient solution compared to onboard processing. This scheme guarantees that computational tasks are handled efficiently without overburdening the vehicle's onboard resources, thereby extending the vehicle's battery life and maintaining system performance. For the vehicular task, the offloading decision is denoted as $\xi_{I_{t,k}} \in \{0,1\}$. And $\xi_{I_{t,k}} = 1$ if the task $I_{t,k}$ is offloaded to the RSU. Conversely, if the task is assigned to the subsequent RSU along the vehicle's path, $\xi_{I_{t,k}} = 0$. This decision-making process is intricate and considers various factors such as the current server load, task priority, and vehicle mobility to optimize resource allocation effectively. Furthermore, the strategy considers the dynamic nature of vehicular networks, adjusting the offloading decisions as the vehicle moves. The task execution delay on the MEC server is determined by several factors, including the processing power of the MEC server and the communication delays between the vehicle and the server. The latency for executing task on MEC is calculated as:

$$T_{I_{t,k}}^{r} = \frac{\left(1 - p_{I_{t,k}}\right) D_{I_{t,k}}}{f^r} \tag{6}$$

where f^r represents the MEC's CPU frequency.

2.4 Overall computing model

Task offloading causes latency that comprise of four parts, the uploading delay, the waiting time, the time spent waiting for the server processing time, the execution time of the task and the downloading time of the results. The outcomes of the process are of smaller size compared to the task; thus, the delay and energy consumption is negligible as highlighted in [3]. Additionally, the task latency and the transmission of results between RSUs are also negligible since the bandwidth of fiber is much greater than that of wireless channels [13]. Therefore, the total delay of the task offloading is denoted as:

$$T_{I_{t,k}}^{V2I}\left(\xi_{I_{t,k}}\right) = T_{I_{t,k}}^{up} + \xi_{I_{t,k}}\left(T_{I_{t,k}}^{r} + T_{I_{t,k},r}^{wait}\right) + \left(1 - \xi_{I_{t,k}}\right)\left(T_{I_{t,k}}^{r+1} + T_{I_{t,k},r+1}^{wait}\right) \tag{7}$$

where the task queue time is denoted by $T^{wait}_{I_{t,k},r}$. Subsequently the task is executed on-vehicle and offloaded to the server simultaneously, the overall processing delay is determined by the longer duration between $T^{V2I}_{I_{t,k}}$ and $T^{local}_{I_{t,k}}$, as follows:

$$T_{I_{t,k}} = \begin{cases} T^{local}_{I_{t,k}} & \text{if } p_{I_{t,k}} = 1 \\ \max\left\{T^{local}_{I_{t,k}}, p^{V2I}_{I_{t,k}}\left(\xi_{I_{t,k}}\right)\right\} & \text{if } 0 < p_{I_{t,k}} < 1 \\ p^{V2I}_{I_{t,k}}\left(\xi_{I_{t,k}}\right) & \text{if } p_{I_{t,k}} = 0 \end{cases} \tag{8}$$

Hence, the total energy utilization for the task $I_{t,k}$ is given by the sum of the energy used for V2I communication $E^{V2I}_{I_{t,k}}$ and the energy used for local processing $E^{local}_{I_{t,k}}$. This combined metric provides a comprehensive view of the energy demands associated with task execution, factoring in both offloading and local computation components.

$$E_{I_{t,k}} = E^{V2I}_{I_{t,k}} + E^{local}_{I_{t,k}} \tag{9}$$

3 PROBLEM FORMULATION

We intend to minimize both delay and energy consumption for the task offloading problem. Therefore, our objective function is formulated as the weighted sum of latency and energy consumption, as expressed follows:

$$\sum\nolimits_{I_{t,k}\in I} U_{I_{t,k}} = \sum\nolimits_{I_{t,k}\in I} \left(\alpha \mathrm{T}_{I_{t,k}} + (1-\alpha)\omega E_{I_{t,k}}\right) \tag{10}$$

where ω serves as a normalizing factor, allowing the two terms to be combined without units. Furthermore, the objective function, along with the constraints, is represented below:

$$\begin{aligned} & \min \sum_{I_{t,k}\in I} U_{I_{t,k}} \\ s.t.\quad & \mathrm{C1}: T_{I_{t,k}} \le T^{\max}_{I_{t,k}} \forall I_{t,k} \in I \\ & \mathrm{C2}: \sum_{I_{t,k}\in I} E_{I_{t,k}} \le E^{k} \forall k \in K, \\ & \mathrm{C3}: 0 \le p_{I_{t,k}} \le 1 \quad \forall I_{t,k} \in I \\ & \mathrm{C4}: \xi_{I_{t,k}} \in \{0,1\} \quad \forall I_{t,k} \in I \end{aligned} \tag{11}$$

Where C1 denotes that the task latency should not exceed the specified threshold, C2 suggests that the total energy consumption of the vehicle k must be less than the tolerable energy. The constraint C3 indicates the constraints on the tasks that are locally executed, while C4 specifies that the task $I_{t,k}$ is either offloaded to the next RSU along the vehicle's route or to the nearby RSU. These constraints are crucial for maintaining efficient task processing and energy management within the system.

4 PROPOSED SCHEME

In this section, we present a solution to the task offloading problem using the DRL-PPO algorithm. The problem is modeled as a MDP, where states, actions, and the reward function are clearly defined to enable efficient learning and decision-making.

4.1 Markov decision process model

1. **State space:** The state at any given time t for all vehicles is denoted as $s_t = \left\{s_t^1, s_t^2, \ldots, s_t^k\right\}$, where s_t^k represents the state of vehicle k. Each vehicle's state includes critical real-time information required for the task offloading decision. Formally, the state at time t can be represented as:

$$S_t = \left\{I_{t,k}, B_k^r, R_k^r, f^r\right\} \tag{12}$$

Where $I_{t,k}$ denotes the task assigned to vehicle k at time t, B_k^r indicates the available bandwidth, R_k^r specifies the transmission rate, and f^r is the CPU frequency of MEC server.

2. **Action space:** The action space defines the possible actions that can be taken at each state. These actions include the decisions regarding the proportion of the task to offload to the MEC server or process locally, and the selection of the target MEC server for offloading. Formally, the action at time\(t \) can be represented as:

$$A_t = \left\{p_{I_{t,k}}, \xi_{I_{t,k}}\right\} \tag{13}$$

where $p_{I_{t,k}}$ is the proportion of the task processed locally, $\xi_{I_{t,k}}$ is offloading decision, where 0 indicates offloading to the next RSU and 1 indicates offloading to the nearest RSU.

3. **Reward function:** The reward function evaluates the effectiveness of the action taken in each state. It aims to balance the trade-off between minimizing energy consumption and reducing task processing latency. The reward R_t at time t can be formulated as:

$$R_t = -\left(\alpha T_{I_{t,k}} + (1-\alpha)\omega E_{I_{t,k}}\right) \tag{14}$$

where $T_{I_{t,k}}$ is the total task processing time, $E_{I_{t,k}}$ is the total energy consumption, and (α) is a weight factor balancing the importance of latency and energy consumption.

4.2 Algorithm based on deep reinforcement learning-proximal policy optimization

We begin by initializing the policy parameters θ and the value function parameters ϕ. An empty replay buffer is also initialized to store transitions. For each episode, the initial state S_0 is initialized, where $S_0 = \left\{I_{0,k}, B_k^r, R_k^r, f^r\right\}$ representing the task information, bandwidth, data transmission rate, and CPU frequency of the MEC server.

At each time step t, an action $A_t = \left\{p_{I_{t,k}}, \xi_{I_{t,k}}\right\}$ is selected based on the current policy $\pi_\theta\left(A_t | S_t\right)$. The action determines the fraction of the task processed locally and the offloading decision to either the nearest or the next RSU. The action is executed, and the next state S_{t+1} and reward R_t are observed. The reward $R\_t = -(\alpha T_{I_{t,k}} + \left(1-\alpha\right)\theta\, E_{I_{t,k}}$ is designed to minimize the total energy consumption and latency of task processing.

The transition (S_t, A_t, R_t, S_{t+1}) is stored in the replay buffer. After collecting sufficient transitions, a batch is sampled from the replay buffer. The advantages $\widehat{A}_t$ are computed using generalized advantage estimation (GAE). The clipped surrogate objective is optimized to update the policy parameters or theta:

$$L^{CLIP}(\theta) = \widehat{E}_t\left[\min\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}\widehat{A}_t, \text{clip}\left(\frac{\pi_\theta(A_t|S_t)}{\pi_{\theta_{old}}(A_t|S_t)}, 1-\epsilon, 1+\epsilon\right)\widehat{A}_t\right)\right] \tag{15}$$

The policy parameters θ are updated using gradient ascent:

$$\theta \leftarrow \theta + \alpha\nabla_\theta L^{CLIP}(\theta) \tag{16}$$

Concurrently, the value function parameters ϕ are updated by minimizing the squared error loss:

$$L^{VF}(\phi) = \widehat{E}_t\left[\left(V_\phi(S_t) - \widehat{R}_t\right)^2\right] \tag{17}$$

The value function parameters ϕ are updated using gradient descent:

$$\phi \leftarrow \phi - \beta\nabla_\phi L^{VF}(\phi) \tag{18}$$

The process is repeated for a predefined number of episodes or until convergence, ensuring the policy learns to optimize task offloading effectively.

Algorithm 1: DRL-PPO Algorithm

```
Initialization:
  - Initialize the parameters for both the policy (θ) and the value function (ϕ).
  - Initialize an empty replay buffer to store transitions.
For each episode:

    Initialize the starting state $S_0$, where $S_0 = \{I_{0,k}, B_k^r, R_k^r, f^r\}$,
      For each time step t:
        - Select action $A_t = \{p_{I_{t,k}}, \xi_{I_{t,k}}\}$ based on the current policy $\pi_\theta(A_t|S_t)$.
        - Execute the action $A_t$, observe the next state $S_{t+1}$\) and reward $R_t$.
        - Store the transition in the replay buffer ($S_t$, $A_t$, $R_t$, $S_{t+1}$).
        - Update the state $S_t = S_{t+1}$.
      End
      After collecting sufficient transitions:
        - Sample a batch of transitions from the replay buffer.
        - Compute the advantages $\widehat{A}_t$ using GAE.

      Optimize the policy:
      - Compute the clipped surrogate objective function to ensure stable policy updates.
      - Update the policy parameters θ using gradient ascent to maximize the objective.

      Optimize the value function:
      - Compute the loss, according to (17).
      - Update the value function according to (18).
End
- Repeat the process for a predefined number of episodes or until the policy converges.
```

5 NUMERICAL RESULTS

In this section, we performed a simulation to evaluate the effectiveness of our system. The simulation covered an 800-meter two-way network, managed by four RSUs, each with a 200-meter coverage and integrated by the MEC server. Upon entry from various points, vehicles followed a poisson process. Each vehicle generated a task, and the neighboring RSU was alerted to organize the proposed offloading method. The proposed offloading approach determined the target execution methods applied on the partitioned four segments for each task. It is divided into tasks in size and delay thresholds for routing, traffic scheduling, and video tasks. Navigation tasks were measured at a size of 3 MB and an acceptable delay of 400 ms, traffic scheduling tasks were measured at a size of 0.5 MB and a delay threshold of 150 ms. Video tasks had a size of 10 MB and a delay threshold of 1500 ms. Moreover, the parameters used in our simulation comprise Brk = 20 MHz for the allocated bandwidth, transmission power Prk = 0.5 W, and CPU frequencies fk = 4 G cycles/s for the vehicles and fr = 10 G cycles/s for the MEC servers. The Gaussian noise power is set at $\sigma = 10^{-13}$ W, while the energy consumption coefficient per CPU cycle is $K = 10^{-26}$ J/Cycles. The values for p range from 0 to 1 in increments of 0.25. Other parameters include $\alpha = 0.8$ and $\omega = 0.1$. These settings ensure a comprehensive evaluation of our proposed system.

To evaluate the performance more effectively, we compare our proposed DRL-PPO algorithm with other benchmark algorithms: actor-critic (AC) and deep Q-network (DQN). It is hence useful in drawing a comparison that will help reveal the benefits that the DRL-PPO approach brings in terms of efficiency and improvements gained.

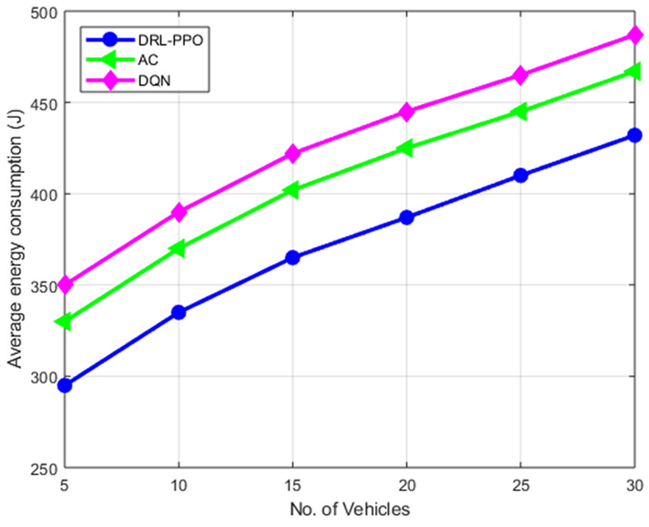

Fig. 1. Average task energy

Figure 1 shows the average energy consumption for all algorithms, i.e., DRL-PPO, AC, and DQN. The results imply that the average energy consumption increases proportionally with the number of vehicles. Among all the algorithms, DRL-PPO exhibits the lowest energy consumption, indicating its superior efficiency. AC shows slightly higher energy consumption, while DQN presents the highest energy consumption that aligns with expectations. It can be observed that as the number of vehicles increases, all algorithms show a rise in energy consumption. These findings underscore the potential of DRL-PPO in energy-efficient applications, such as EVs, where it can enhance system performance, extend battery life, and provide an effective solution for managing energy consumption and delay constraints in vehicular networks.

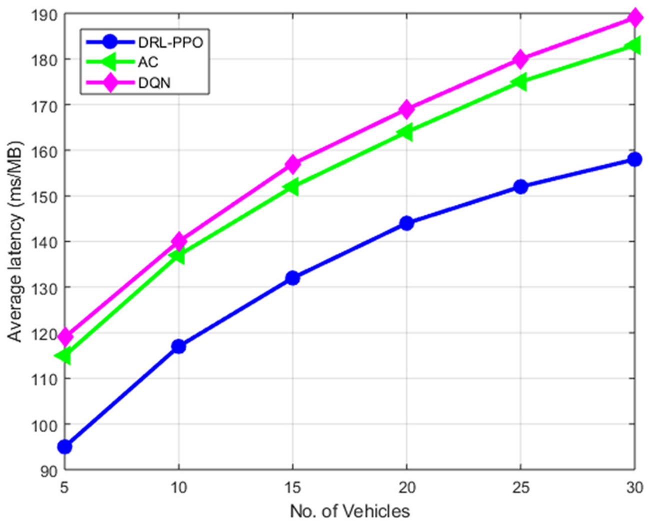

Fig. 2. Average task latency

Figure 2 shows the average latency where three algorithms, which include DRL-PPO, AC, and DQN, are depicted with the increase in the number of vehicles ranging from 5 to 30. As can be demonstrated for the measure of latency, DRL-PPO is always the lowest one, demonstrating superior efficiency. Both AC and DQN exhibit higher latencies, with DQN performing the worst. As vehicle numbers rise, latency increases for all algorithms, but DRL-PPO maintains a clear advantage. This efficiency highlights DRL-PPO's effectiveness in real-time, latency-sensitive applications in vehicular networks, making it ideal for enhancing performance and the user experience.

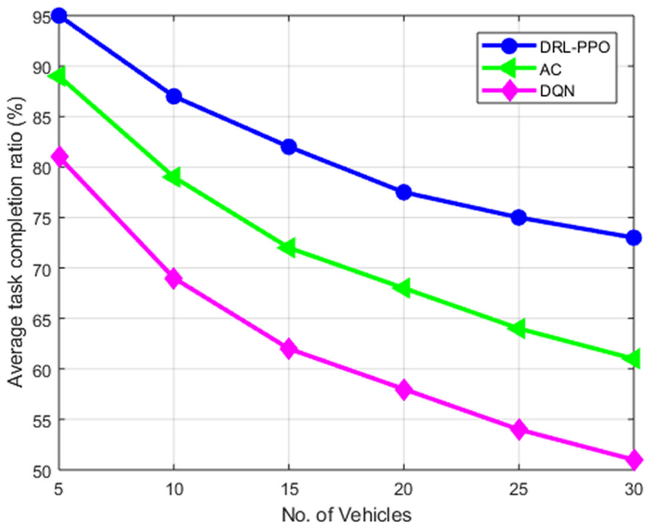

Fig. 3. Average task completion rate

Figure 3 depicts the plots of the task completion rates in the case of all the algorithms involving the increase in the number of vehicles from five to 30. However, DRL-PPO is able to achieve a maximum task completion ratio over time and outperforms or is competitive with other methods with an increasing number of vehicles. AC completes slightly more problems, while DQN solves minimal of all the problems. This is because as the number of vehicles increases, the probability of completing tasks reduces as resources are restricted and competition is evident from the statistics observed by all the algorithms. The results presented demonstrate that DRL-PPO outperforms the basic DRL in terms of managing the network's resources, therefore, its usage is relevant in cases where high reliability and performance are expected in vehicular networks.

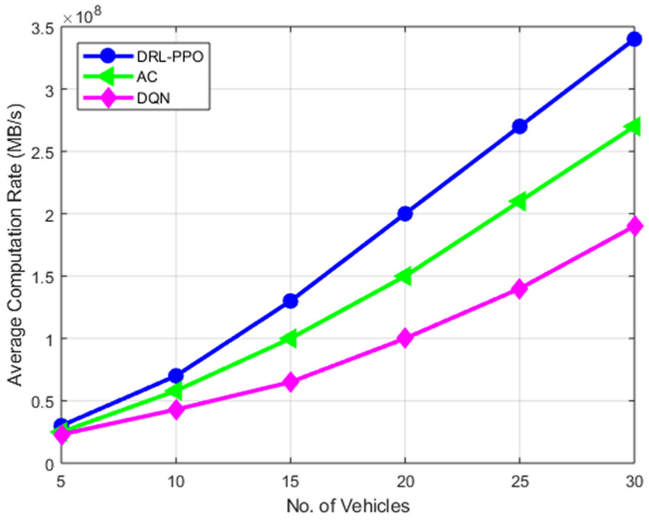

Fig. 4. Average computation rate

Figure 4 illustrates the average computation rate of all algorithms displayed against the number of vehicles, varying from five to 30. As it can be observed from Figure 4, that the results associated with the computation rate indicate that DRL-PPO remains the top-performing agent due to better optimization of computational resources. AC works in a missionary fashion, whereas DQN work the least in terms of computation rates. With an increase in vehicles, computation rates increase for all the algorithms, while DRL-PPO surpasses the others by a considerable margin. This highlights DRL-PPO's effectiveness in managing and optimizing computational tasks in vehicular networks.

6 CONCLUSION

This study examined the difficulties that arise for electricity-powered intelligent connected vehicles in handling the growing number of computational activities. Through the utilization of MEC and the introduction of a task offloading approach that takes into account the movement of users, we enhance the efficiency of allocating computational resources from servers located at the roadside. The use of our mathematical model, represented as MDP, together with the implementation of the DRL-PPO algorithm exhibits noteworthy decreases in energy usage and latency when compared to other benchmark DRL techniques. This strategy improves the effectiveness and productivity of job execution in electric cars, therefore contributing to the progress of the automotive industry. In the future, the author intends to examine the scalability and implementation of the suggested job offloading system in bigger networks with a larger number of vehicles and roadside servers, encompassing both urban and rural environments in real-world scenarios. By integrating the scheme with 6G technology, the latency may be further reduced, and the communication reliability between EVs and MEC servers can be enhanced.

7 REFERENCES

[1] H. Bagheri *et al.*, “5G NR-V2X: Toward connected and cooperative autonomous driving,” *IEEE Communications Standards Magazine*, vol. 5, no. 1, pp. 48–54, 2021. https://doi.org/10.1109/MCOMSTD.001.2000069

[2] M. Muratori *et al.*, “The rise of electric vehicles—2020 status and future expectations,” *Progress in Energy*, vol. 3, no. 2, p. 022002, 2021. https://doi.org/10.1088/2516-1083/abe0ad

[3] S. Raza, S. Wang, M. Ahmed, M. R. Anwar, M. A. Mirza, and W. U. Khan, “Task offloading and resource allocation for IoV using 5G NR-V2X communication,” *IEEE Internet of Things Journal*, vol. 9, no. 13, pp. 10397–10410, 2021. https://doi.org/10.1109/JIOT.2021.3121796

[4] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, “A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions,” *Wireless Communications and Mobile Computing*, vol. 2019, no. 1, pp. 1–19, 2019. https://doi.org/10.1155/2019/3159762

[5] J. Liu *et al.*, “McVCO: Multi-MEC cooperative vehicular computation offloading,” *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 813–826, 2023. https://doi.org/10.1109/TIV.2023.3299381

[6] S. Raza, M. Ahmed, H. Ahmad, M. A. Mirza, M. A. Habib, and S. Wang, “Task offloading in mmWave based 5G vehicular cloud computing,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 12595–12607, 2023. https://doi.org/10.1007/s12652-022-04320-y

[7] M. Ahmed *et al.*, “A survey on vehicular task offloading: Classification, issues, and challenges,” *Journal of King Saud University – Computer and Information Sciences*, vol. 34, no. 7, pp. 4135–4162, 2022. https://doi.org/10.1016/j.jksuci.2022.05.016

[8] S. Raza *et al.*, “An efficient task offloading scheme in vehicular edge computing,” *Journal of Cloud Computing*, vol. 9, no. 1, 2020. https://doi.org/10.1186/s13677-020-00175-w

[9] M. Qin *et al.*, “Service-oriented energy-latency tradeoff for IoT task partial offloading in MEC-enhanced multi-RAT networks,” *IEEE Internet of Things Journal*, vol. 8, no. 3, pp. 1896–1907, 2020. https://doi.org/10.1109/JIOT.2020.3015970

[10] C. Li, K. Jiang, Y. Zhang, L. Jiang, Y. Luo, and S. Wan, “Deep reinforcement learning-based mining task offloading scheme for intelligent connected vehicles in UAV-aided MEC,” *ACM Transactions on Design Automation of Electronic Systems*, vol. 29, no. 3, pp. 1–29, 2024. https://doi.org/10.1145/3653451

[11] S. J. Ankar and K. P. Pinkymol, “Optimal sizing and energy management of electric vehicle hybrid energy storage systems with multi-objective optimization criterion,” *IEEE Transactions on Vehicular Technology*, vol. 73, no. 8, pp. 11082–11096, 2024. https://doi.org/10.1109/TVT.2024.3372137

[12] L. Xu, H. Tang, H. Li, X. Li, T. A. Gulliver, and K. N. Le, “Secrecy performance intelligent prediction for mobile vehicular networks: An DI-CNN approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 7, pp. 7363–7373, 2024. https://doi.org/10.1109/TITS.2024.3352668

[13] M. Ahmed *et al.*, “Active reconfigurable intelligent surfaces: Expanding the frontiers of wireless communication-A survey,” *IEEE Communications Surveys & Tutorials*, 2024. https://doi.org/10.1109/COMST.2024.3423460

8 AUTHOR

Salman Raza is with the Department of Computer Science, National Textile University Faisalabad, Faisalabad, Punjab, Pakistan (E-mail: salmanraza@ntu.edu.pk).