Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Dynamic Task Offloading in Vehicular Edge Networks: A Mobility-Driven A3C-Based Framework

Manoj Kumar Somesula

manojsomesula@nitj.ac.in

Dr. B. R. Ambedkar National Institute of Technology Jalandhar

Research Article

Keywords: Vehicular edge computing, Deep Reinforcement learning, A3C, User mobility, offloading scheme

Posted Date: July 7th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-8148128/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Noname manuscript No.
(will be inserted by the editor)

Dynamic Task Offloading in Vehicular Edge Networks: A Mobility-Driven A3C-Based Framework

Manoj Kumar Somesula[1]

Abstract Vehicular edge computing (VEC) has emerged as a promising paradigm for reducing response time and alleviating bandwidth congestion in Internet of Vehicles applications. The rapid proliferation of smart and intelligent devices has accelerated the demand for delay-sensitive and computationally intensive services, such as autonomous driving, immersive online gaming, and holographic communication. These applications impose stringent necessities, typically requiring end-to-end latency between 10 and 100 ms, bandwidth ranging from 1 Mbps to 1 Gbps, and high reliability. Meeting these requirements remains a critical challenge, particularly when vehicles traverse across multiple roadside unit coverage regions, where efficient utilization of distributed edge resources and dynamic task offloading strategies are essential. In this study, we investigate the problem of mobility-aware and deadline-constrained task scheduling in vehicular networks. The task offloading problem is modeled as a Markov decision process, and a deep reinforcement learning framework is proposed to optimize scheduling decisions under mobility and latency constraints. Specifically, we adopt the Asynchronous Advantage Actor-Critic (A3C) algorithm, which effectively adapts to vehicular mobility dynamics while minimizing task completion delay and energy overhead. Simulation results demonstrate that the proposed A3C-based approach outperforms existing mobility-aware scheduling strategies, achieving superior performance in terms of delay reduction, energy consumption, and successful task offloading percentage. These findings highlight the potential of deep reinforcement learning in addressing the challenges of large-scale task offloading in VEC.

Keywords Vehicular edge computing · Deep Reinforcement learning · A3C · User mobility · offloading scheme

Manoj Kumar Somesula
E-mail: manojsomesula@nitj.ac.in

[1] Department of Computer Science and Engineering, Dr. B. R. Ambedkar National Institute of Technology Jalandhar, Punjab, 144008, India

1 Introduction

With the rapid advancement of the Internet of Vehicles (IoV), vehicular edge computing (VEC) has emerged as a promising paradigm for distributing computational workloads among vehicles. The proliferation of mobile applications has significantly intensified the demand for powerful computational support within wireless networks [1]. In response, vehicular applications are increasingly being developed to address diverse aspects of daily life, including driver safety, in-vehicle entertainment, and intelligent traffic management [2]. Intelligent transportation systems (ITS) further facilitate the integration of these applications into real-world mobility scenarios [3]. However, such applications' massive volumes of traffic data generate considerable strain on networks and lead to longer processing delays, particularly for vehicles with limited onboard resources [3]. To address these challenges, VEC introduces computation closer to end users, enabling reduced energy consumption and faster task execution. The architecture of VEC leverages distributed computing resources across cloud, edge, and terminal layers, allowing vehicles to offload tasks dynamically according to their specific service requirements [4].

In vehicular edge computing systems, task scheduling within the Internet of vehicles remains a significant challenge. Each vehicle generates a dynamic set of computational requests that are typically offloaded to the nearest road side unit for execution [2]. Inefficient

scheduling mechanisms can considerably degrade the overall performance of VEC networks. Although task scheduling in IoV has been widely explored, limited research has addressed the influence of vehicle mobility and the associated sojourn time within RSU coverage [5]. Incorporating knowledge of vehicular mobility patterns can substantially improve the effectiveness of scheduling strategies. However, when a vehicle transitions from one RSU coverage area to another after submitting its tasks, the ongoing requests may fail, requiring retransmission to the next RSU. The limited computational capacity and low resource capacity of vehicle-mounted modules for real-time data processing, networking, and decision making pose challenges. It emphasises the prevalence of computation and resource-intensive applications, which strain the resource-limited vehicular network.

The aforementioned challenges motivate the exploration of a mobility-aware task scheduling framework within vehicular edge computing environments, where RSUs offload tasks to vehicles for execution. In this context, the RSU is responsible for determining the proportion of tasks that should be processed locally by the RSU and those that should be delegated to vesicles based on the mobility of the vehicle's. Existing studies on mobility-aware offloading have largely focused on simplified assumptions, such as constant velocities, unidirectional single-path mobility, or mobility restricted to a single RSU region [6, 7]. In practice, however, vehicular movement is highly dynamic, as vehicles may vary their speeds due to traffic conditions and traverse multiple paths with bi-directional or multi-directional mobility. Consequently, efficient task scheduling requires accurate estimation of both the dwell time of vehicles within an RSU's coverage and their potential travel paths in the base station's service region. Leveraging such predictive mobility information is essential for optimizing resource allocation and enhancing the overall performance of VEC networks.

Vehicular applications often generate computation-intensive requests, leading to significant energy consumption in on-board processors. When tasks are offloaded to vehicles. Consequently, there is a pressing need for a task allocation mechanism that is both delay-aware and energy-efficient to meet the stringent requirements of Internet of Vehicles applications. In this study, we address the limitations of existing vehicular edge computing task scheduling approaches by incorporating task deadlines and vehicular mobility information into the scheduling framework. The mobility of vehicles is modeled using a Markov renewal process (MRP), enabling accurate prediction of sojourn times and travel paths. To overcome this challenge, we propose a deep reinforcement learning algorithm named A3C, ensuring efficient and scalable mobility-aware task scheduling in large-scale vehicular edge environments.

- Develop an Integer Linear Programming (ILP) formulation for the task offloading problem in vehicular edge networks, aiming to maximize the overall utility while satisfying constraints on computation capacity, task deadlines, and vehicle sojourn times.
- Formulate the intelligent task scheduling problem as an ILP that maximizes per-task utility, rewarding timely completion of tasks while penalizing energy consumption. Additionally, model the problem as a Markov Decision Process (MDP) to optimize the cumulative reward by coordinating vehicle actions over time.
- Design a task offloading scheme using a deep reinforcement learning framework, specifically the Asynchronous Advantage Actor-Critic (A3C) algorithm, to accelerate learning and improve task scheduling efficiency at the vehicle level, thereby maximizing overall utility.
- Conduct extensive simulations to evaluate the performance and effectiveness of the proposed deep reinforcement learning-based task offloading framework, considering multiple performance metrics such as latency, energy consumption, and task completion rate.

2 Related Work

The work [8], presents an energy-efficient multi-relay scheduling on deployed RSU powered by alternative energy sources. The study depicts the reduction in power consumption of the roadside unit in green vehicular networks via proper scheduling from the RSU to the vehicle [8]. The work [9], focuses on the vehicle's task processing latency and energy consumption by deciding that the task should be scheduled to either the local RSU or neighbouring RSU along the vehicle path. In this study, they address the deadlines of tasks and mobility information of vehicles to reduce the energy consumption of vehicles and the latency of vehicular computational tasks, which are not present in existing vehicular fog computing VFC task scheduling strategies [10].

The study [5], depicts that vehicular edge computing can cache contents in different roadside units at the network edge to support the real-time vehicular networks . Here, the paper considers that user data contains private information, users are reluctant to share their data with others, to solve this problem, traditionally, federated learning is used to protect user privacy

[5]. The research formulates a joint optimisation problem related to distributed task offloading and multi-resource management; optimisation aims to maximise the system utility by creating an optimal policy for task and resource scheduling [2]. The research seeks to improve the response latency in vehicular networks [2]. The work focuses on caching popular content at the base stations; the base stations work together to cache content that the users likely request [11]. The approach avoids the need to fetch content from the remote servers, which directly reduces the latency [11]. The research study [12], focused on improving data traffic management in wireless networks through content caching techniques. The main goal is to reduce user data retrieval delays by storing popular files locally. Since the popularity of content can change over time, the research explores ways to adapt caching strategies to these variations [12].

The framework presented in the work leverages nearby edge nodes to promptly process emergency information and services, enhancing service quality [13]. The paper also introduces a joint task offloading and edge caching problem to optimize service latency and vehicle energy consumption [13]. The study addresses the issue of traditional offloading decisions, made based on local vehicle states, which often fail to fully utilize the mobile edge computing (MEC) server resources and typically overlook vehicle mobility for simplicity, this study introduces a DRL based task offloading decision algorithm called DROD (Deep Reinforcement learning based Offloading Decision) for vehicular edge computing, DROD here considers the vehicle mobility and signal blocking to minimize the system overhead [14] Two main research challenges discussed in the study are ensuring consistent access to cloud services regardless of the operating system or hardware used, and maintaining security and privacy during data transmission and offloading [15]. The paper proposes a system that trains a model to update edge devices' information, enhancing the 6G-V2X network's communication framework and addressing key optimization challenges. This system aims to improve the overall efficiency of edge cloud systems by providing updated information crucial for model optimization [15].

2.1 Mobility-sensitive scheduling

The proposed solution involves distributing mobile delay-sensitive tasks over multiple fog-based base stations within the fog layer of an SDN-enabled architecture [16]. This approach leverages the mobility of vehicles to optimize data distribution and transmission, thereby improving network flexibility, controllability, and the overall success rate in addressing the multiple time-constrained vehicular applications scheduling issue [16]. The work in the offloading process considers network delay, flow-rule capacity, and link utilization, while the downloading process incorporates vehicle mobility alongside these parameters [17]. This study examines the design of mobility-aware computation offloading for mobile edge computing in vehicular wireless networks, considering the random mobility of vehicles and the potential for handovers during the offloading process. The offloading decisions are made by comparing the average total costs of executing computation tasks locally on the vehicle versus remotely on MEC servers [18].

2.2 Task-centric scheduling

Task-centric scheduling is vital in fog computing for vehicular networks as it ensures the efficient offloading and processing of tasks by considering each task's specific requirements and characteristics. This approach helps manage the high mobility of vehicles and the uneven distribution of vehicle density, preventing performance issues such as unbalanced workload and unexpected task failures. This article [19], explores a new service scenario of task offloading within a three-layer service architecture, leveraging the resources of fog server, vehicular fog and central cloud in a cooperative manner. The work [20], addressed the challenge where the limited computing capacity of vehicles restricts the support for these compute-intensive applications, hindering the development of intelligent connected vehicles. To address the proposed challenge, vehicular edge computing networks integrate mobile edge computing with vehicular networks[20]. Deploying mobile edge computing servers at the network edge significantly reduces the computational burden on ICVs through MEC offloading. An intelligent task offloading scheme based on deep Q-learning is designed to solve the problem. To overcome these challenges, [20] design an intelligent task offloading scheme based on deep Q-learning.

Delay-conscious scheduling is crucial in addressing the demands of delay-sensitive vehicular applications, such as automatic driving and auto navigation, within vehicular edge computing networks. This scheduling approach ensures that tasks are offloaded to edge servers efficiently by minimizing processing delays for vehicles' compute-intensive tasks. The study proposes the online unmanned aerial vehicle approach to manage vehicle edge computing overload, which minimizes vehicular task delays while adhering to UAV constraints [21].

3 System Model and Problem Formulation

In this section, the network model and problem formulation are presented in detail.

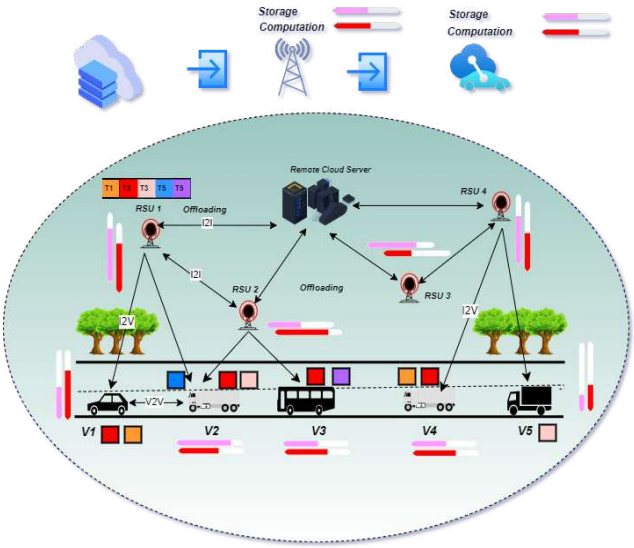


Fig. 1 Network Model

3.1 Network Model

We envision a vehicular edge network in rural areas, as depicted in Fig. 1, where a limited number of roadside units are strategically placed along roads within a smart village, overseen by a central base station. The RSUs are spread across the rural landscape, deliberately positioned at significant intervals to maximize coverage without overlap. This placement strategy recognizes the vast expanses characteristic of rural areas, ensuring that each RSU serves its designated area effectively while optimizing resources.

In this work, we consider vehicular network contemplate of V mobile vehicles, set of road side units R, and the cloud is deployed to associate with RSUs and vehicles. The set of V vehicles designated as $\mathcal{V} = \{1, 2, \ldots, v, \ldots, V\}$ with communication, limited computational and storage abilities ready to extend the services to latency-sensitive IoT applications, a set of R RSUs represented as $\mathcal{R} = \{1, 2, \ldots, r, \ldots, R\}$ and a cloud. These set of mobile vehicles are distributed over the area considered. The vehicles are directly connected to the associated RSU and the vehicle will not be in coverage of more than one RSU. Each of the RSU covers some set of vehicles. The RSUs are connected to the central cloud via wired medium. Vehicles are linked to the nearby road side units developing vechicle to infrastructure communication. Vehicles are connected to nearby vehicles in their range, developing vehicle-to-vehicle communication. Vehicles can directly communicate to the cloud server establishing the vehicle to cloud communication.

Task offloading from vehicles to roadside units involves direct real-time communication between RSU and the vehicles. When a vehicle approaches an RSU, the RSU assesses the vehicle's available computational resources and the nature of the task. If the vehicle is capable of handling the task, the RSU transmits the necessary data and instructions directly to the vehicle. This process relies on low latency communication protocols to ensure the timely task execution and completion. Since there is no caching involved, the data is processed on-the-fly, requiring efficient and reliable connectivity between the RSU and the vehicle to maintain performance and avoid delays in the model.

3.2 Mobility Model

RSUs send the task to vehicles which are moving from one RSU coverage to another RSU coverage in specific time slot. Vehicle mobility is modeled using a Markov renewal process, $\{(X_g, T_g) : g \geq 0\}$, to predict IoV trajectories and their dwell times at roadside units (RSUs), where X_g denotes the g^{th} transition state and T_g the corresponding transition time. The **sojourn time**, Δ_v^r, represents the duration a vehicle v remains within the coverage of RSU r, derived from mobility statistics. The transition probabilities of the Markov chain are given by P_v, and H_v^r represents the distribution of time a vehicle spends at RSU r before moving to another. computing environments.

3.2.1 Path and Sojourn Time Prediction

Over a time interval T, a vehicle v transitions between different states, and the exact duration of its stay within an RSU r is unknown. The sojourn time, Δ_v^r, representing the time a vehicle remains within the coverage of RSU r, is estimated using a Markov Renewal Process [10, 22]. The expected sojourn duration is computed as the average of the predicted dwell times, expressed as follows:

$$\mu_v^r = \int_0^\infty y H_v^r(y)\, dy \tag{1}$$

$H_v^r(y)$ denotes the probability density function for the sojourn time of the vehicle v in a state r, i.e., the vehicle is expected to travel from one RSU to another by staying μ_v^r average time duration in RSU r.

Vehicles may move along multiple directions within a given time interval T. The trajectory of a vehicle v along a path can be represented as $\mathbf{Z} = z_v^1, z_v^2, \ldots, z_v^k$, where $k \geq 1$ denotes the number of paths. The initial state of the k^{th} path is denoted by $z_v^{k,0} = z_v^k$, indicating that the starting state is common across all paths of vehicle v. Consequently, the k^{th} path of vehicle v can be expressed as $z_v^k = z_v^{k,0}, z_v^{k,1}, \ldots, z_v^{k,b}$, where $b \geq 0$ and $z_v^{k,1}$ corresponds to the first RSU encountered along this path. A vehicle may visit the same RSU multiple times along a given path, making z_v^k a multi-set. The vehicle remains in the initial state for a duration of t_v^0 before making the first transition. For all paths, the timing of this first transition is expected to occur at the following time instant:

$$\mathrm{T}_v^{k,1} = \int_0^\infty y H_v^{z_v^{k,0}}(y)\, dy \tag{2}$$

Where $\mathrm{T}_v^{k,0}$ is 0 and $\mathrm{T}_v^{k,1}$ is constant for all the paths of the vehicle v. Therefore, the k^{th} path j^{th} transition is expected to be happened as follows:

$$\mathrm{T}_v^{k,j} = \mathrm{T}_v^{k,j-1} + \mu_v^{z_v^{k,j-1}} \tag{3}$$

where $j <= b$. The final transition occurs at the end of time instant T.

The sojourn time of the k^{th} path z_v^k is represented as $\Delta_v^k = \{\Delta_v^{z_v^{k,0}}, \Delta_v^{z_v^{k,1}}, ..., \Delta_v^{z_v^{k,b}}\}$. The sojourn time is $\Delta_v^{z_v^{k,j}} = \mathrm{T}_v^{k,j} - \mathrm{T}_v^{k,j-1}$. Hence, the sojourn time is elaborated as:

$$\Delta_v^{z_v^{k,j}} = \begin{cases} \mathrm{T}_v^{k,1}, & j = 0 \\ [\mu^r_{z_v^{k,j}}], & 1 <= j <= b-1 \\ [\mathrm{T} - \sum_{w=0}^{b-1} \Delta_v^{z_v^{k,w}}], & j = b \end{cases} \tag{4}$$

Where $q = [\mathrm{u}]$ denotes the value $q = 0$ when $\mathrm{u} < 0$ and $q = \mathrm{u}$ when $\mathrm{u} >= 0$. The probability of a vehicle v choosing the path z_v^k is defined as:

$$\mathrm{P}_v^k = \prod_{j=0}^{b-1} \mathrm{P}_{z_v^{k,j}, z_v^{k,j+1}} \tag{5}$$

Therefore, The average sojourn duration of vehicle v at RSU r during the time period T is denoted as:

$$\Delta_v^r = \sum_{k \in K} \mathrm{P}_v^k \sum_{w=0}^{d} \Delta_v^{z_v^{k,j}} 1_{(z_v^{k,w} = r)}, \forall r \in \mathcal{R} \tag{6}$$

3.3 Task Model

In the vehicular edge computing environment, each RSU $r \in \mathcal{R}$ generates a set of computational tasks $\mathcal{T}_r = \{1, 2, \ldots, T_r\}$. Each task $t \in \mathcal{T}_r$ is described by a set of parameters representing its data requirements, computational load, deadline, and criticality: $\mathcal{T}_{r,t} = \{d_{r,t}, c_{r,t}, \delta_{r,t}, \}$, where: $d_{r,t}$ characterises the size of input data (in bits), $c_{r,t}$ symbolises the number of CPU cycles demanded, and $\delta_{r,t}$ is the maximum allowable time for task completion (deadline) of the task. For each RSU $r \in \mathcal{R}$ and task $t \in \mathcal{T}_r$, let $x_{r,t} \in \{0,1\}, x_{r,t}^v \in \{0,1\}\ \ \forall v \in \mathcal{V}$, where $x_{r,t} = 1$ selects local execution at RSU and $x_{r,t}^v = 1$ selects offloading to vehicle v. Each task can be executed either on the RSU or offloaded to vehicle $v \in \mathcal{V}$.

3.3.1 Local Execution

Tasks executed on the RSU utilize its processor with computing capability c_r. RSU execution incurs no transmission delay but consumes the RSU's energy. The execution delay is calculated as:

$$D_{r,t}^{loc} = \frac{c_{r,t}}{c_r}. \tag{7}$$

The energy consumption for local processing is modeled as:

$$E_{r,t}^{loc} = \kappa c_r^2 c_{r,t}, \tag{8}$$

where κ is the effective switched capacitance coefficient.

3.3.2 Vehicle Offloading

Tasks can be offloaded to vehicles v for execution. Offloading introduces both transmission and execution delays:

$$D_{r,t}^v = \frac{d_{r,t}}{B_{r,v}} + \frac{c_{r,t}}{c_r}, \tag{9}$$

where $B_{r,v}$ is the available uplink bandwidth, and c_r is the RSU's computational capacity. Energy consumed during offloading is:

$$E_{r,t}^{tx} = P_{tx} \cdot \frac{d_{r,t}}{B_{r,v}}, \tag{10}$$

where P_{tx} is the RSU's transmission power.

The total delay and energy for task t are expressed as:

$$D_{r,t} = x_{r,t} D_{r,t}^{loc} + \sum_{v \in \mathcal{V}} x_{r,t}^v D_{r,t}^v, \tag{11}$$

$$E_{r,t} = x_{r,t} E_{r,t}^{loc} + \sum_{v \in \mathcal{V}} x_{r,t}^v E_{r,t}^{tx}, \tag{12}$$

where $x_{r,t}$ and $x_{r,t}^v$ are binary indicators for local execution and vehicle offloading, respectively.

Define a per–task utility that rewards meeting deadlines and penalizes energy:

$$U_{r,t} = w_d \underbrace{\max\Big\{0,\, 1 - \tfrac{D_{r,t}}{\delta_{r,t}+\epsilon}\Big\}}_{\text{deadline satisfaction}} - w_e \underbrace{\frac{E_{r,t}}{E_{\max}}}_{\text{normalized energy}}, \tag{13}$$

where $w_d, w_e > 0$ are weights, $E_{\max} > 0$ normalizes energy, and $\epsilon > 0$ avoids division by zero.

3.4 Problem Formulation

Maximizing the overall utility guarantees that tasks are executed within their respective deadlines while simul-

taneously reducing the energy consumption of both vehicles and roadside units.

$$\max_{\{x_{r,t},x^v_{r,t}\}} \sum_{r\in\mathcal{R}}\sum_{t\in\mathcal{T}_r} U_{r,t}. \tag{14}$$

3.5 Task Scheduling Constraints

Assignment constraint:

$$x_{r,t} + \sum_{v\in\mathcal{V}} x^v_{r,t} = 1, \qquad \forall r \in \mathcal{R},\ \forall t \in \mathcal{T}_r. \tag{15}$$

Deadline constraint:

$$D_{r,t} \le \delta_{r,t}, \qquad \forall r, t. \tag{16}$$

Sojourn-time constraint: If vehicle v is currently under RSU r with sojourn time Δ^r_v, and for the predicted path $\mathcal{P}_v$ from the MRP,

$$x^v_{r,t}\, D^{\mathrm{v}}_{r,t}(r) \le \Delta^r_v, \qquad \forall r, t,\ \forall r \in \mathcal{P}_v. \tag{17}$$

Computation constraint: Let e_r denote the per-slot compute budget of RSU r (in CPU-cycles per slot):

$$\sum_{t\in\mathcal{T}_r} x_{r,t}\, c_{r,t} \ \le\ c_r, \qquad \forall r \in \mathcal{R}. \tag{18}$$

similarly, compute budget of Vehicle v is:

$$\sum_{t\in\mathcal{T}_r} x^v_{r,t}\, c_{v,t} \ \le\ c_v, \qquad \forall v \in \mathcal{V}. \tag{19}$$

Decision variables:

$$x_{r,t} \in \{0,1\}, \quad x^v_{r,t} \in \{0,1\}, \qquad \forall v, t, r. \tag{20}$$

The task scheduling formulation in Eq. (14) represents an integer linear programming (ILP) problem, which has been demonstrated to be NP-hard [10, 22]. Solving this problem requires determining the optimal set of decision variables $\{X_t\}$ at each time slot. Since the evaluation of Eq. (14) involves the distribution of a summation of independent random variables, the computation becomes non-trivial. Moreover, the decision variables in Eq. (20) are integer-valued and vary dynamically, thereby necessitating the collection of extensive network state information. This challenge is further compounded by the exponential growth in the number of mobile users, which leads to prohibitive computational overhead when attempting direct optimization of the objective function. Due to factors such as vehicular mobility, randomness in connectivity, and the need for adaptive decision-making, conventional optimization approaches [22] are unsuitable for this problem. Recent advancements in deep learning, with its strong representation and generalization capabilities [22, 23], have motivated the use of learning-based approaches in wireless networks. Building on this insight, we develop a deep reinforcement learning framework to intelligently and cooperatively schedule task offloading in Internet of Vehicles (IoV) environments.

4 Deep Reinforcement Learning mechanism

To address the proposed scheduling problem, we develop a reinforcement learning (RL)-based framework in which an intelligent agent learns to derive an optimal caching and scheduling strategy within the mobile edge network. RL enables the agent to interact with the dynamic environment, adapt through trial-and-error, and refine its policy based on past experiences and feedback Fig. 2. The primary goal of the RL agent is to maximise the cumulative reward by effectively balancing exploration of possible actions and exploitation of the most beneficial ones. In this study, the system states are defined by temporal factors and mobile devices, while the actions correspond to task allocation or caching decisions. The reward function is formulated in terms of delay reduction, capturing the efficiency of the scheduling process. Since the set of available devices changes dynamically due to mobility, the action space cannot be predefined. To overcome this, we extend the framework into a deep reinforcement learning (DRL) approach, enabling the agent to generalize over varying environments and select the most suitable devices based on expected reward. The base station assists this process by maintaining contextual information such as the number of neighboring devices, average contact durations, transmission rates, and storage capacities, which supports more accurate decision-making.

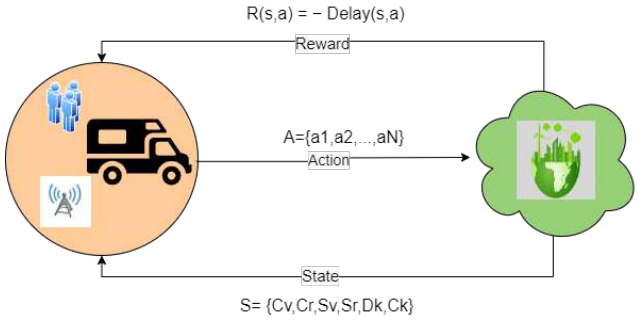


Fig. 2 RL Environmental cycle

4.1 Task Scheduling using Deep Reinforcement Learning

Conventional optimization methods are inadequate for handling the highly dynamic, complex, and uncertain characteristics of vehicular networks, particularly in scenarios involving computation offloading and resource allocation. To address these challenges, DRL approaches have appeared as promising alternatives. Among them, advanced techniques such as Mixed Integer Linear Programming (MILP) and the Double Deep Q-Network

(DDQN) have been widely explored. Specifically, DDQN has demonstrated superior performance for vehicular offloading tasks, as it can effectively adapt to fluctuating and unpredictable network conditions. Unlike static optimization frameworks, DDQN is capable of continuously learning from the environment, updating its policies, and making near real-time decisions. This adaptability allows it to efficiently manage system complexity, scale with increasing network size, and enhance decision-making through sustained interactions with the environment, thereby ensuring robust and efficient task scheduling in dynamic vehicular scenarios.

Asynchronous Advantage Actor-Critic (A3C) (shown in Fig. 3) is superior for offloading in vehicular networks compared to double deep Q-network and traditional optimization techniques due to several reasons. A3C uses multiple parallel agents for faster learning and adapts efficiently to dynamic conditions. It provides stable learning through actor-critic networks and continuously adapts to changes. A3C manages resources effectively, maintains a good balance between exploration and exploitation, and scales well with network size and complexity. These features make A3C particularly suited for the fluctuating and complex environment of vehicular networks. To formulate the problem as a reinforcement learning task, we need to define the state space, action space, and reward function.

The task scheduling and offloading problem in vehicular networks can be formulated as a Markov Decision Process (MDP), defined by the tuple: $M = (S, A, P, R, \gamma)$

State Space The state space represents the environment's current state, which includes information necessary for decision making. In this problem, the state can include; Each state $s \in \mathcal{S}$captures the current status of the vehicular network, including: vehicle locations and velocities, tasks queued at each vehicle, available computational resources at RSUs, and sojourn times of vehicles within RSU coverage. The state may also incorporate historical observations to enhance policy learning in a deep reinforcement learning framework. Each state s can be represented as a tuple $(C_v, C_r, S_v, S_r, D_k, C_k)$ where C_v and C_r represent the available computational capacities of vehicles and RSUs, respectively, S_v and S_r represent the available storage capacities of vehicles and RSUs, respectively. D_k represents the data size of task k, C_k represents the computational requirements of task k.

Action Space The action space defines the possible actions that the agent (vehicles or RSUs) can take in a given state. In this problem, the action space can include: Offloading a task to a specific vehicle or RSU and Processing a task locally by a vehicle. Let's denote the action space as A= $(a_1, a_2, \ldots, a_n)$ where each action a_i corresponds to offloading the task to a specific entity or processing it locally.

Reward Function The reward function defines the immediate reward received by the agent after taking an action in a certain state. In this problem, the objective is to maximization the total delay time during task offloading. Therefore, a suitable reward function could be:

$$R(s, a) = w_d D_{r,t} - w_e E_{r,t} \tag{21}$$

$D_{r,t}$ is the Deadline Reward whic is Positive reward if the task finishes before its deadline. $E_{r,t}$ is the Energy Cost that is the Normalized energy consumption of local computation or offloading and w_d, w_e are weights balancing latency and energy. The RL agent aims to learn a policy π that maximizes the cumulative reward over time by interacting with the environment.With these components defined, a reinforcement learning algorithm such as deep Q-networks or deep deterministic policy gradient can be applied to learn optimal task offloading strategies in vehicular networks.

4.2 Workflow

The sequence of steps that outlines the workflow of the proposed A3CRL algorithm is described below.

Initialization:
The framework in algorithm 1, begins by initializing global parameters for the policy network (θ) and the value network (θ_v), which are crucial for decision-making and value estimation. A shared counter (T) is initialized to track global training steps, with a maximum step limit (T_{max}) set to control training duration. Shared memory is defined for experience replay, enabling workers to store and retrieve past experiences during learning. Each worker(vehicle/RSU) is equipped with local parameters (θ' and θ'_v) synchronized with the global network.

Worker Function:
Each worker operates independently within a loop, starting with the initialization of a local counter (t), environment, and initial state (s). The worker repeats actions based on the policy $\pi(a|s;\theta')$, collects rewards, and stores state transitions in an experience buffer. The state value function $V(s_t;\theta_v)$, that parameterised by θ_v is denoted as

$$V(s_t;\theta_v) = E[G_t \mid s = s_t, \pi] = E\Big[\sum_{k=0}^{\infty} \gamma^k r_{t+m} \mid s = s_t, \pi\Big] \tag{22}$$

here G_t denotes the discouted return for the state s_t and γ depicts future returns of the present state. At terminal states or when reaching the maximum steps per episode (T_{max}), the worker computes advantages

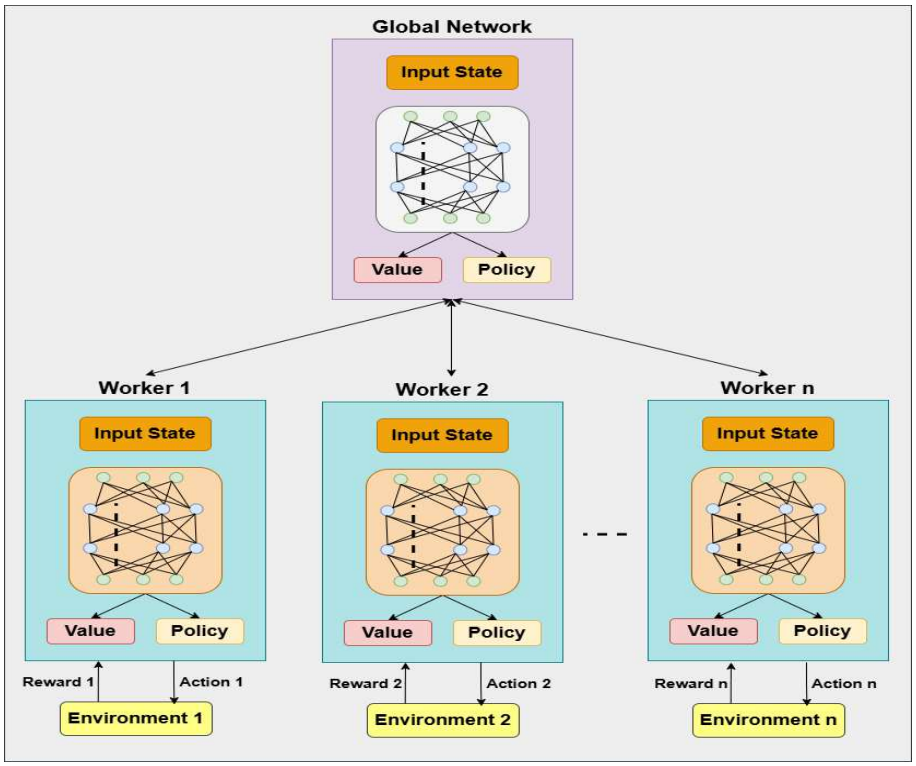


Fig. 3 The architecture of A3C

and updates gradients for both the policy and value networks asynchronously. This involves accumulating gradients with respect to the policy parameters (θ) and value parameters (θ_v) based on computed advantages and policy gradients.

Utility Function:
Utility functions like Policy Network and Value Network define the forward passes of the policy and value networks respectively. These functions compute action probabilities and state values based on current network parameters.

Main Execution:
In the main execution block, multiple workers are created and started concurrently. Each worker engages in its training process independently, executing episodes of interactions with the environment, updating parameters based on computed gradients, and leveraging experience replay for efficient learning. The main thread supervises the workers, ensuring synchronization and completion of all training episodes before concluding the learning process.

This framework facilitates decentralized learning in vehicular networks, optimizing task offloading decisions based on real-time state information and reinforcement learning principles, thereby enhancing system efficiency and performance in dynamic environments. Adjustments can be made to adapt the framework to specific network conditions and application requirements.

5 Performance Evaluation

We assess the performance of the proposed method through comprehensive simulations conducted in a realistic vehicular network setting. The findings indicate that our scheme delivers a higher task-completion success rate—expressed as the successful offloading rate and reduces energy consumption compared with existing solutions. For benchmarking, A3CRL is contrasted with four representative task-scheduling strategies: (i) DDPG [24] (ii) DDQN [25] (iii) SA-PSO [9] and (iv) GA [26]. The evaluation employs several performance indicators, including the average service delay—capturing both transmission time to neighbouring resources and RSUs, and computation time, the average energy consumed by vehicles (in joules), and the successful offloading of tasks, which together reflect the objective function presented in Eq. 14.

5.1 Simulation Environment

We assess the performance of the proposed A3CRL algorithm using Python implementations executed in

Algorithm 1 Mobility Aware Offloading using A3C Reinforcement Learning

Initialize global parameters for policy network (θ) and value network (θ_v)
Initialize shared counter $T = 0$
Set maximum global steps $T_{\max}$
Define shared memory for experience replay
function WORKER(id)
    Initialize local counter $t = 0$
    Initialize environment and state s
    repeat
        Reset gradients: $d\theta = 0$ and $d\theta_v = 0$
        Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
        Initialize experience buffer
        for $t_{\text{local}} = 1$ **to** $T_{\max}$ **do**
            Perform action a based on policy $\pi(a|s;\theta')$
            Receive reward r and new state s'
            Store (s, a, r, s') in experience buffer
            if s' is terminal **or** $t_{\text{local}} == T_{\max}$ **then**
                $R = 0$ **if** s' is terminal, **else** $R = V(s', \theta'_v)$
                for each experience (s, a, r, s') in reverse order **do**
                    $R = r + \gamma \cdot R$
                    Compute advantage $A = R - V(s, \theta'_v)$
                    Accumulate gradients with respect to θ: $d\theta = d\theta + \nabla_{\theta'} \log \pi(a|s;\theta') \cdot A$
                    Accumulate gradients with respect to θ'_v: $d\theta_v = d\theta_v + \nabla_{\theta'_v}(R - V(s;\theta'_v))^2$
                end for
                Perform asynchronous update of global parameters using $d\theta$ and $d\theta_v$
                Clear experience buffer
                Reset environment and state s
            end if
            $s = s'$
            $t = t + 1$
            $T = T + 1$
            if $T > T_{\max}$ **then**
                Exit loop
            end if
        end for
    until $T > T_{\max}$
end function
function POLICY NETWORK(state, θ)
    // Define the forward pass of the policy network
    return action probabilities
end function
function VALUE NETWORK(state, θ_v)
    // Define the forward pass of the value network
    return state value
end function
procedure MAIN
    // Create and start multiple workers
    for $i = 1$ **to** number_of_workers **do**
        Start Worker(i)
    end for
    Wait for all workers to finish
end procedure

Table 1 Simulation Parameters

Parameters	Values
Vehicles Speed (km/h)	50
Number of vehicles	50
Noise power N_0	-100 dB
Vehicle CPU frequency	5 GHz
Edge server CPU frequency	15 GHz
Data size of the task	250-500 KB
Computation cycles of the task(10^9)	0.5-5
Transmission power of RSU server	50 dbM
Transmission power of Vehicle	20-25 dbM
RSU server bandwidth	30-50 MHz
Allowable task deadline	0.1-0.4 s

a Jupyter Notebook environment [10]. The simulation parameters and their corresponding values are summarized in Table 1. The vehicular edge computing scenario includes between 3 and 10 RSUs, and 10 to 50 vehicles equipped with onboard processors, following the settings in [27]. RSUs are randomly positioned within the network, each connected to a base station. Vehicle speeds range from 10 to 50 km/h, while CPU frequencies for vehicles and RSU edge servers are set between 2–5 GHz and 10–15 GHz, respectively. Task characteristics include data sizes of 250–500 KB and computational demands between 0.5–5 Giga cycles. Transmission power levels are fixed at 50 dBm for RSUs and 20–25 dBm for vehicles, and the available bandwidth at each RSU is assumed to be 30–50 MHz. Task deadlines fall within 0.1–0.4 seconds.

The setup used for implementing the DDPG and DDQN task-scheduling models is summarized in Table 1. For DDPG, both the actor and critic networks consist of two hidden layers with 500 neurons each and are trained using a batch size of 64. The replay memory is configured to store up to 20,000 transitions, and the learning rate for all network updates is fixed at 0.005. A softmax activation function is employed, the discount factor is set to 0.9, and training is carried out for 4,000 episodes. For DDQN, the Q-network and target network follow an identical hidden-layer structure, use the same batch size and replay-buffer capacity, and are optimized with the same learning rate and discount factor. Both models are trained and tested within the specified VEC environment, and each experimental trial is repeated 50 times. The final performance results represent the average over all runs, and the proposed approach is evaluated against contemporary state-of-the-art baseline algorithms under equivalent simulation settings.

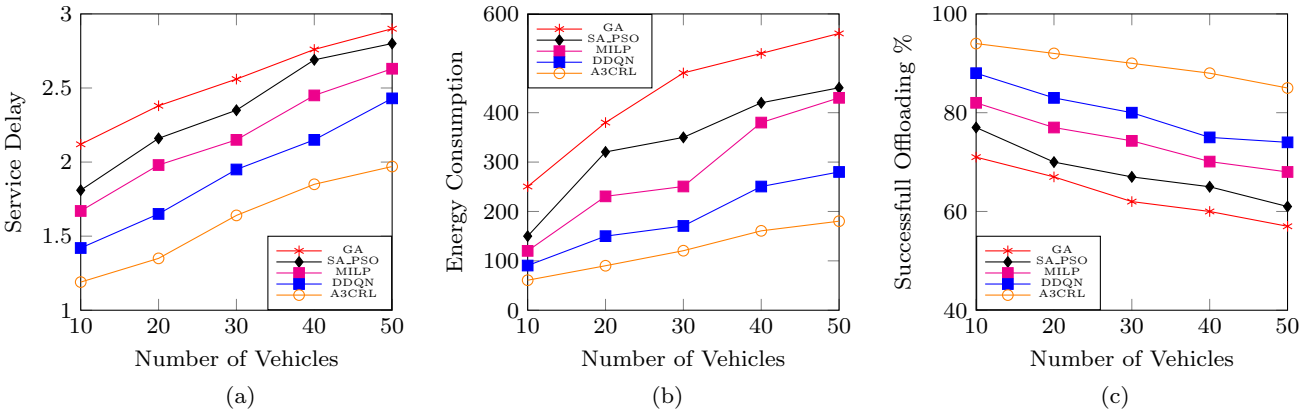


Fig. 4 Impact of the proposed A3CRL and existing algorithms performance with number of vehicles vs (a) Service Delay (b) Energy Consumption (c) Successfully offloaded task percentage

5.2 Reference Algorithms

To demonstrate the efficiency of the proposed method, we compare it to the following task offloading approaches in this section: SA-PSO [9], DDPG [24], DDQN [25] and GA [26].

- Simulated Annealing Particle Swarm Optimization (SA-PSO) [9]: SA-PSO combines Simulated Annealing (SA) and Particle Swarm Optimization (PSO) techniques. SA mimics the annealing process to escape local minima, while PSO simulates the social behavior of particles to explore and exploit the search space effectively. This hybrid approach aims to enhance optimization performance and convergence in various applications.
- The Deep Deterministic Policy Gradient (DDPG) method employs an off-policy actor–critic framework in which the actor generates continuous control actions and the critic estimates their value for the current state. To enhance stability during training, it incorporates slowly updated target networks and reuses stored transitions from a replay buffer, allowing effective learning in continuous-action settings.
- Greedy Algorithm (GA) [26]: Greedy algorithms make locally optimal choices at each step with the hope of finding a global optimum. They are straightforward to implement but may not always yield the best solution for complex optimization problems.
- Double Deep Q-Network (DDQN): DDQN extends deep Q-networks by using two neural networks — one for action selection (policy network) and one for action evaluation (target network). It addresses Q-learning's overestimation bias to improve stability and learning efficiency in reinforcement learning tasks.

5.3 Impact of Number of Vehicles

In this subsection, the proposed A3CRL task-scheduling strategy is analysed by varying the vehicular density from 10 to 50 vehicles per kilometre, while maintaining a constant speed of 50 km/h, as illustrated in Fig. 4. As shown in Fig. 4a, an increase in the number of vehicles results in a rise in task service delay. This is expected because a higher vehicular density increases the computational load on each node, as vehicles must process both their own tasks and those offloaded by neighbouring vehicles. Despite this, the proposed A3CRL algorithm consistently achieves shorter service times compared with competing approaches. By assigning tasks based on vehicular mobility patterns and social proximity, A3CRL is able to distribute tasks more effectively, resulting in up to a 19% reduction in service delay over the second-best method. Fig. 4b shows the trend in per-vehicle energy consumption as vehicular density increases. Although energy usage naturally rises with more vehicles in the network, the A3CRL algorithm maintains a significantly lower consumption level due to its explicit energy-aware objective function. Overall, A3CRL achieves up to a 35% improvement in energy efficiency in comparison with the next strongest baseline. The probability of successful task completion, shown in Fig. 4c, also decreases as the number of vehicles grows. Nonetheless, the uccessful task completion achieved by A3CRL remains consistently higher than that of the benchmark algorithms. Because the proposed utility function incorporates deadline constraints, A3CRL preferentially selects vehicles with stable communication links and adequate computational resources, enabling timely task execution. This results in an improvement of up to 13% when compared to the second-best strategy.

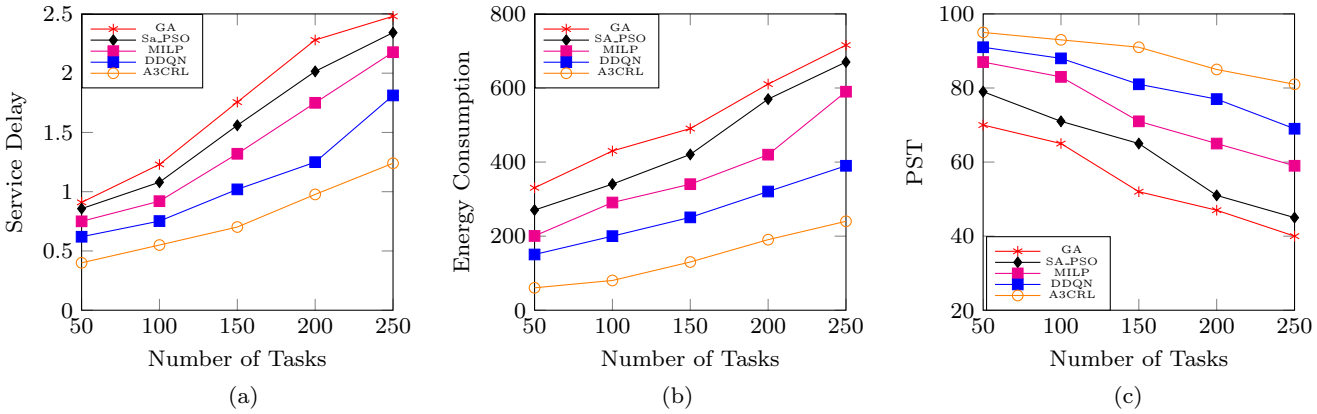


Fig. 5 Impact of the proposed A3CRL and existing algorithms performance with Number of Tasks vs (a) Service Delay (b) Energy Consumption (c) Quality of Service (d) PST

5.4 Impact of Task

In this subsection, the performance of the proposed A3CRL task-scheduling framework is assessed by varying the number of tasks from 50 to 250, while maintaining a vehicular density of 20 vehicles per kilometre travelling at 50 km/h, as illustrated in Fig. 5. The evaluation focuses on analysing how A3CRL responds to increasing computational demand in the vehicular edge computing environment. As depicted in Fig. 5a, the service time naturally increases as the task load rises, owing to the additional processing burden placed on each vehicle. Despite this increase, A3CRL consistently achieves a lower service delay than the benchmark schemes. This improvement stems from its ability to distribute tasks across neighbouring vehicles by leveraging mobility characteristics and social affinity among nodes. Overall, A3CRL demonstrates up to a 29% reduction in service delay relative to the next best method. Similarly, Fig. 5b presents the average energy consumption per vehicle as the number of tasks grows. A gradual increase in energy usage is expected with a higher computational load. However, the A3CRL strategy yields significantly lower energy consumption due to its task-allocation mechanism, which incorporates vehicular sojourn time and social connectivity to reduce unnecessary processing and communication overhead. The results indicate that A3CRL achieves up to a 40% improvement in energy efficiency compared to the second-best approach.Finally, Fig. 5c illustrates the probability of successful task completion, which decreases with increasing task load. Even under heavier workloads, the successful task completion achieved by A3CRL remains higher than that of the comparison algorithms. This advantage results from the design of A3CRL, in which the success rate is embedded within the optimization objective, enabling it to assign tasks to vehicles with more stable communication links and adequate computation resources. The proposed scheme yields an improvement of up to 10% over the second-best result.

5.5 Impact of Vehicle Speed

In this subsection, the performance of the proposed A3CRL task-scheduling algorithm is examined under varying vehicular speeds ranging from 30 to 70 km/h, while maintaining a constant network configuration of 30 vehicles and 150 tasks, as illustrated in Fig. 6. As shown in Fig. 6a, task service latency increases with higher vehicle speeds due to reduced contact duration and greater instability in vehicular links. Nevertheless, the proposed A3CRL approach consistently achieves lower service delay than the benchmark algorithms. By assigning tasks to neighbouring vehicles with favourable mobility characteristics and shorter expected processing times, A3CRL effectively mitigates latency, achieving up to a 19% improvement over the second-best method. Fig. 6b presents the total energy consumption of vehicles as speed increases. Although energy usage rises with higher mobility, the proposed scheme maintains significantly lower consumption levels. This improvement results from A3CRL's objective function, which incorporates mobility-aware energy optimisation and allocates tasks to vehicles capable of completing them with reduced energy expenditure. Overall, A3CRL achieves up to a 35% reduction in average energy consumption compared to the closest competing approach. Fig. 6c illustrates the probability of successful task completion, which decreases as vehicular speed increases. Even under these challenging conditions, A3CRL maintains the highest successful task completion among the evaluated algorithms. Since the success rate is embed-

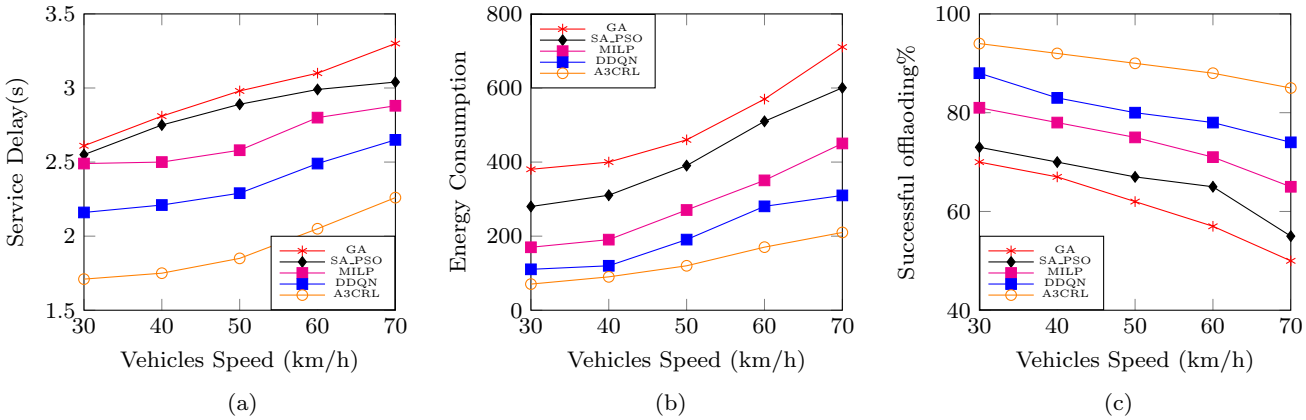


Fig. 6 Proposed A3CRL and existing algorithms performance with Vehicles Speed vs. (a) Service Delay (b) Energy Consumption (c) Successful Offloading percentage

ded in the utility function, the algorithm systematically selects vehicles with reliable communication and computational capability, thereby enhancing task completion within deadline constraints. The proposed method achieves up to a 12% improvement in successful task completion compared with the next strongest competitor.

The baseline task-scheduling algorithms considered in this study exhibit several inherent limitations that restrict their effectiveness in vehicular edge computing environments. The DDQN approach employs a greedy allocation strategy, which becomes inefficient as the network scales and, moreover, disregards the relationships among vehicles. This omission leads to reduced task-completion rates and lower successful task offloading percentage. In contrast, the DDPG method incorporates both mobility and social similarity but overlooks the sojourn duration of vehicles relative to their neighbouring peers, a factor that directly influences task execution reliability. Additionally, its emphasis on social similarity alone fails to capture soujourn time and mobility, thereby weakening communication reliability and contributing to higher task failures, increased service time, elevated energy consumption, and a consequent decline in successful task offloading percentage.

The proposed A3CRL framework addresses these shortcomings by integrating key contextual factors—including vehicle mobility, inter-vehicle social relationships, resource availability, and task deadlines—into its scheduling process. Nevertheless, revisiting the limitations of the baseline methods further highlights the advantages of A3CRL. The DDPG algorithm [24], while mobility-aware, neglects essential relationship between users, resulting in increased task failures and greater delays, underscoring the need for a more holistic solution that accounts for mobility dynamics in VEC systems. Similarly, although DDQN [25] includes mobility, it does not evaluate vehicle sojourn time, which is critical for ensuring timely task completion. SA-PSO task scheduling further burdens individual vehicles by forcing them to process all tasks onboard, causing longer execution times and reducing overall system efficiency. Meanwhile, GA task scheduling disregards both task requirements and resource constraints, resulting in suboptimal task distribution and reduced network performance. Collectively, these limitations contribute to higher service delays, increased energy consumption, and diminished successful task offloading percentage, as demonstrated in Fig. 5, Fig. 4, and Fig. 6.

5.6 Convergence of proposed A3CRL

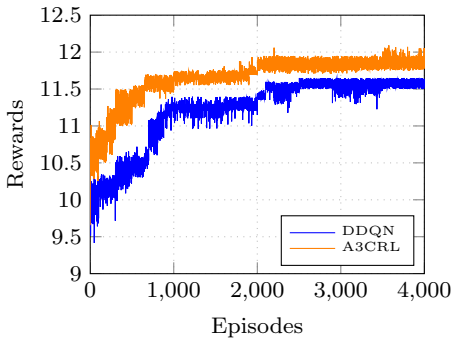


Fig. 7 Convergence comparison of proposed A3CRL

The convergence behaviour of the proposed A3CRL algorithm is illustrated in Fig. 7. For evaluation, A3CRL is compared with the deep deterministic policy gradient (DDQN) method using the reward function defined. The DDQN approach, a model-free deep reinforcement learning algorithm that improves upon the standard DQN by decoupling action selection from action evaluation to reduce overestimation bias, serves as the baseline [10]. The results indicate that DDQN converges

more slowly than the proposed A3CRL strategy. The mean and standard deviation of the reward values obtained through A3CRL are 11.64 and 0.39, respectively, whereas DDQN yields a mean reward of 11.21 with a standard deviation of 0.46. The higher average reward attained by A3CRL demonstrates its superior learning effectiveness throughout the training process. In addition, the lower variance observed in A3CRL indicates that the reward values remain more tightly concentrated around the mean, signifying improved stability and consistency in learning compared with DDQN. The enhanced exploratory capability of the A3CRL framework within the VEC environment further contributes to achieving higher reward outcomes.

6 Conclusion

This work presents a mobility and sojourn time aware task offloading scheme in VEC. Initially, the task offloading problem is mathematically modeled as Integer linear programming problem to maximize the energy consumption and successful task offloading rate of the vehicles. Further, the problem is modeled as a Markov decision process, capturing vehicle mobility, dynamic network conditions, and task deadlines. The proposed method intelligently assigns tasks to either local execution or vechicles, optimizing a utility function that balances latency and energy efficiency, with mobility prediction via a Markov renewal process further improving decision accuracy. The proposed deep reinforcement learning framework, A3C, for mobility-aware task scheduling in vehicular edge computing networks is comprehensively evaluated using baseline algorithms. Simulation results indicate that the framework outperforms conventional optimization techniques, offering a scalable, adaptive, and efficient solution for real-time task scheduling. Future work will investigate multi-objective optimization, cooperative offloading, and adaptive learning strategies, alongside experimental validation to enhance robustness and practical applicability in dynamic vehicular environments.

Declarations

Conflict of interest statement: The authors declare that they do not have any known competing interest.
Funding information: Not applicable.
Informed consent: Not applicable.
Ethical statement: The work submitted by the authors is his own work and it is neither published nor considered for publication elsewhere.
Data availability statement: Code availability: The program of this paper is supported by custom code. It can be applied from the corresponding author on reasonable request.

References

1. Salman, Raza1 W, Liu2* M, Ahmed3 M, Anwar1 M, Ayzed, Mirza4 Q, Sun1 S, Wang1 (2020) An efficient task offloading scheme in vehicular edge computing. Journal of Cloud Computing DOI 10.1186/s13677-020-00175-w
2. Liu L, Feng J, Mu X, Pei Q, Lan D, Xiao M (2023) Asynchronous deep reinforcement learning for collaborative task computing and on-demand resource allocation in vehicular edge computing. IEEE Transactions on Intelligent Transportation Systems 24(12):15513–15526, DOI 10.1109/TITS.2023.3249745
3. Lv Z, Lou R, Singh AK (2021) Ai empowered communication systems for intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems 22(7):4579–4587, DOI 10.1109/TITS.2020.3017183
4. Liu S, Liu L, Tang J, Yu B, Wang Y, Shi W (2019) Edge computing for autonomous driving: Opportunities and challenges. Proceedings of the IEEE 107(8):1697–1716, DOI 10.1109/JPROC.2019.2915983
5. Wu Q, Zhao Y, Fan Q, Fan P, Wang J, Zhang C (2023) Mobility-aware cooperative caching in vehicular edge computing based on asynchronous federated and deep reinforcement learning. IEEE Journal of Selected Topics in Signal Processing 17(1):66–81, DOI 10.1109/JSTSP.2022.3221271
6. Raza S, Wang S, Ahmed M, Anwar MR, Mirza MA, Khan WU (2021) Task offloading and resource allocation for iov using 5g nr-v2x communication. IEEE Internet of Things Journal
7. Misra S, Bera S (2019) Soft-van: Mobility-aware task offloading in software-defined vehicular network. IEEE Transactions on Vehicular Technology 69(2):2071–2078
8. Vemireddy S, Rout RR (2020) Clustering based energy efficient multi-relay scheduling in green vehicular infrastructure. Vehicular Communications 25:100251, DOI https://doi.org/10.1016/j.vehcom.2020.100251, URL `https://www.sciencedirect.com/science/article/pii/S221420962030022X`
9. Raju MR, Mothku SK, Somesula MK (2024) Mits: Mobility-aware intelligent task scheduling in vehicular fog networks. IEEE Transactions on Vehicular Technology 73(3):3079–3093, DOI 10.1109/TVT.2023.3321806

10. Raju MR, Mothku SK, Somesula MK (2024) Dmits: dependency and mobility-aware intelligent task scheduling in socially-enabled vfc based on federated drl approach. IEEE Transactions on Intelligent Transportation Systems pp 17007 – 17022
11. Somesula MK, Rout RR, Somayajulu D (2022) Cooperative cache update using multi-agent recurrent deep reinforcement learning for mobile edge networks. Computer Networks 209:108876, DOI https://doi.org/10.1016/j.comnet.2022.108876
12. Zhong C, Gursoy MC, Velipasalar S (2020) Deep reinforcement learning-based edge caching in wireless networks. IEEE Transactions on Cognitive Communications and Networking 6(1):48–61, DOI 10.1109/TCCN.2020.2968326
13. Wu J, Wang J, Chen Q, Yuan Z, Zhou P, Wang X, Fu C (2021) Resource allocation for delay-sensitive vehicle-to-multi-edges (v2es) communications in vehicular networks: A multi-agent deep reinforcement learning approach. IEEE Transactions on Network Science and Engineering 8(2):1873–1886, DOI 10.1109/TNSE.2021.3075530
14. Zhan W, Luo C, Wang J, Wang C, Min G, Duan H, Zhu Q (2020) Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing. IEEE Internet of Things Journal 7(6):5449–5465, DOI 10.1109/JIOT.2020.2978830
15. Hasan MK, Jahan N, Nazri MZA, Islam S, Attique Khan M, Alzahrani AI, Alalwan N, Nam Y (2024) Federated learning for computational offloading and resource management of vehicular edge computing in 6g-v2x network. IEEE Transactions on Consumer Electronics 70(1):3827–3847, DOI 10.1109/TCE.2024.3357530
16. Lin C, Han G, Qi X, Guizani M, Shu L (2020) A distributed mobile fog computing scheme for mobile delay-sensitive applications in sdn-enabled vehicular networks. IEEE Transactions on Vehicular Technology 69(5):5481–5493, DOI 10.1109/TVT.2020.2980934
17. Misra S, Bera S (2020) Soft-van: Mobility-aware task offloading in software-defined vehicular network. IEEE Transactions on Vehicular Technology 69(2):2071–2078, DOI 10.1109/TVT.2019.2958740
18. Huy Hoang V, Ho TM, Le LB (2020) Mobility-aware computation offloading in mec-based vehicular wireless networks. IEEE Communications Letters 24(2):466–469, DOI 10.1109/LCOMM.2019.2956514
19. Liu Z, Dai P, Xing H, Yu Z, Zhang W (2022) A distributed algorithm for task offloading in vehicular networks with hybrid fog/cloud computing. IEEE Transactions on Systems, Man, and Cybernetics: Systems 52(7):4388–4401, DOI 10.1109/TSMC.2021.3097005
20. Guo H, Liu J, Ren J, Zhang Y (2020) Intelligent task offloading in vehicular edge computing networks. IEEE Wireless Communications 27(4):126–132, DOI 10.1109/MWC.001.1900489
21. Dai X, Xiao Z, Jiang H, Lui JCS (2024) Uav-assisted task offloading in vehicular edge computing networks. IEEE Transactions on Mobile Computing 23(4):2520–2534, DOI 10.1109/TMC.2023.3259394
22. Somesula MK, Rout RR, Somayajulu DV (2021) Contact duration-aware cooperative cache placement using genetic algorithm for mobile edge networks. Computer Networks 193:108062
23. Somesula MK, Kotte A, Annadanam SC, Mothku SK (2022) Deadline-aware cache placement scheme using fuzzy reinforcement learning in device-to-device mobile edge networks. Mobile Networks and Applications 27(5):2100–2117
24. Liu G, Dai F, Huang B, Qiang Z, Wang S, Li L (2022) A collaborative computation and dependency-aware task offloading method for vehicular edge computing: a reinforcement learning approach. Journal of Cloud Computing 11(1):68
25. Ullah I, Han YH (2025) Optimizing vehicular edge computing: graph-based double-dqn approaches for intelligent task offloading. The Journal of Supercomputing 81(1):76
26. Somesula MK, Rout RR, Somayajulu D (2023) Greedy cooperative cache placement for mobile edge networks with user preferences prediction and adaptive clustering. Ad Hoc Networks 140:103051
27. Wei Z, Li B, Zhang R, Cheng X, Yang L (2023) Many-to-many task offloading in vehicular fog computing: A multi-agent deep reinforcement learning approach. IEEE Transactions on Mobile Computing 23(3):2107–2122