Deep Reinforcement Learning for Energy-Efficient Task Offloading in Cooperative Vehicular Edge Networks

Paul Agbaje
dept. Computer Science and Engineering
University of Texas at Arlington
pauloluwatowoju.agbaje@uta.edu

Ebelechukwu Nwafor
dept. Computing Sciences
Villanova University
ebelechukwu.nwafor@villanova.edu

Habeeb Olufowobi
dept. Computer Science and Engineering
University of Texas at Arlington
habeeb.olufowobi@uta.edu

***Abstract*—In the Internet of Vehicle ecosystem, multi-access edge computing (MEC) enables mobile nodes to improve their communication and computation capabilities by executing transactions in near real-time. However, the limited energy and computation capabilities of MEC servers limit the efficiency of task computation. Moreover, the use of static edge servers in dense vehicular networks may lead to an influx of service requests that negatively impact the quality of service (QoS) of the edge network. To enhance the QoS and optimize network resources, minimizing offloading computation costs in terms of reduced latency and energy consumption is crucial. In this paper, we propose a cooperative offloading scheme for vehicular nodes, using vehicles as mobile edge servers, which minimizes energy consumption and network delay. In addition, an optimization problem is presented, which is formulated as a Markov Decision Process (MDP). The solution proposed is a deep reinforcement-based Twin Delayed Deep Deterministic policy gradient (TD3), ensuring an optimal balance between task computation time delay and the energy consumption of the system.**

***Index Terms*—Smart cities, vehicular edge computing, Internet of vehicles, deep reinforcement learning, task offloading, TD3**

I. Introduction

With the increasing reliance on embedded sensors integrated into vehicles and their connectivity to the Internet, the Internet of Vehicles (IoV) has emerged as a network that incorporates the Internet of Things (IoT), communication networks, and the cloud to support vehicular operations. These sensors are an integral aspect of modern automobile architecture, allowing for increased safety, comfort, and automation [1]. In connected autonomous vehicles, big data shared by these sensors have become the ultimate enabler of understanding the virtuous cycle of safety and autonomy and is essential in strengthening the resilience of vehicles in the future connected smart cities.

Currently, cloud computing platforms provide the necessary resources and infrastructures for the efficient operation of IoV and are essential in transforming cars from stand-alone, transportation-centric machines to sophisticated computers on wheels. However, this approach is becoming increasingly inefficient, particularly regarding real-time communication, data processing, and service execution. This inefficiency is due to the voluminous data produced by these embedded sensors, high latency, bandwidth limitations, and potential connectivity failures between the cloud and vehicles. In addition, limited spectrum resources from traditional wireless networks and low computational capabilities of in-vehicle resources pose challenges for real-time and computationally intensive tasks.

To address these challenges, multi-access edge computing (MEC) has been implemented, allowing computation and service delivery to be moved from the cloud to edge nodes. MEC can be deployed at roadside units (RSUs) or base stations (BSs) to minimize network congestion and enhance performance. By offloading delay-sensitive and computationally intensive tasks to the edge network, MEC enables vehicles to improve their communication and computation capabilities and minimize energy consumption [2].

MEC servers provide support for vehicular networks but are limited by cost, scalability, latency, and security issues in meeting the quality of service (QoS) requirements for vehicular applications. Effective task offloading requires vehicles to determine when and what tasks to offload to MEC servers. Energy efficiency and security are crucial in this scenario, where limited network resources must handle numerous task-offloading requests. The increasing number of vehicles and the dynamic nature of the network make it challenging to maintain network efficiency, especially with limited RSUs or BSs. Moreover, a large number of task offloading requests can overload MEC servers, impact QoS, and cause denial of service attacks.

To address these challenges, we propose using vehicles as mobile servers where the vehicles function as mobile nodes providing computing and network services to other vehicles and RSUs. By using the vehicle's resources, MEC servers services can be enhanced, reducing the loads on the servers and improving the overall QoS. As vehicles are also resource-constrained, we leveraged techniques prioritizing tasks based on available computational resources and offloading non-safety-critical but latency-sensitive tasks. This task includes workloads for traffic management, road safety, and navigation in intelligent transportation systems, multimedia processing and streaming for in-vehicle entertainment, and message forwarding between two endpoints. Since the complexity of the problem increases with the mobile environments and actions, we leveraged DRL to develop a Twin Delayed Deep Deterministic policy gradient (TD3)-based algorithm to address the offloading constraints. Simulation results shows the effectiveness of our approach and highlight its potential in supporting MEC for reduced latency and energy efficiency in cooperative vehicular edge networks.

In summary, the contributions of this paper are as follows:

- We propose a vehicular edge computing framework that includes vehicles as edge servers, providing additional computational resources apart from the conventional static edge servers.
- We formulate an optimization problem to minimize the energy consumption rate and total network delay to enhance the QoS in resource-constrained vehicular nodes.
- We reformulate the optimization problem as a Markov decision process (MDP). Due to the complexity of the formulated problem with respect to the complex state, action, and environment, a TD3-based solution is leveraged to develop an algorithm for solving the optimization problem.
- To demonstrate the effectiveness of our approach, we evaluate the performance in a simulated vehicular environment. The result shows the efficiency for task offloading with minimal latency and energy consumption.

II. Related Work

Artificial intelligence (AI) approaches such as reinforcement learning (RL) offer long-term advantages for decision-making in dynamic environments [3]. This has led to the development of RL algorithms for solving offloading problems in vehicular edge networks, such as Q-learning [4], deep Q-learning [5], and deep deterministic policy gradient-based algorithms (DDPG) [6]. Tan and Hu [4] used Q-learning and a multi-time scale framework to address computing resource allocation and caching placement issues in vehicular edge networks with strict service deadlines. Liu et al. [5] applied a DRL-based Q-learning approach to maximize the total utility of a vehicle edge computing network. Hazarika et al. [7] proposed a soft actor-critic (SAC) method for resource allocation during task offloading in IoV, considering task priority and utility functionsto maximize the mean system utility. Peng and Shen [6] developed a hierarchical DDPG algorithm to optimize multi-dimensional resources to satisfy the QoS requirements and maximizing offloaded tasks in vehicular edge networks. However, DDPG policies can sometimes be inaccurate due to incorrect sharp peaks in the Q-function approximator [8].

Despite advancements in DRL algorithms, the influx of requests in the vehicular edge environment can cause processing delay and energy consumption in the network. An efficient approach is required to make offloading decisions and allocate resources while maintaining QoS. To address these challenges, we propose a method that considers the duration of task offloading, computation, and energy consumption. In our proposed model, vehicles can support edge servers by providing computational services. Our offloading scheme optimizes energy efficiency while ensuring QoS for users. Furthermore, the proposed TD3-based algorithm addresses the limitations of DDPG-based algorithms with approaches such as dual Q-function learning, delayed policy updates, and policy smoothing [8].

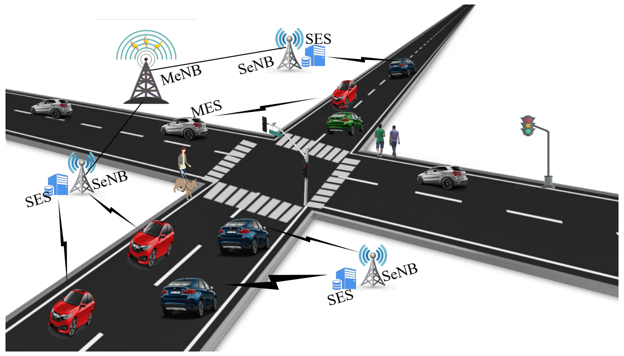

Fig. 1: Overview of the vehicular edge environment

III. System Model

In this section, we present the system model of the vehicular MEC network. We first describe the network model, then describe the details of the communication, computation, and energy models.

A. Vehicular MEC Network Architecture

Fig. 1 illustrates the network architecture of the vehicular MEC network. We consider a vehicular MEC environment with a macrocell (MeNB) denoted by $\mathcal{M}$ and a set of N small cells (SeNBs) denoted by $\mathcal{N} = \{1, 2, 3, ..., N\}$, connected to the MeNB through wired links [9]. The MeNB is connected through the core network to the cloud via cellular communication links and has a MEC server associated with it. Also, a static MEC server (SES) is associated with each SeNB. If the set of vehicles in the network is denoted by $\mathcal{V} = \{1, 2, 3, ..., V\}$, we assume that $|\mathcal{V}| \geqslant |\mathcal{N}| \geqslant |\mathcal{M}|$, represents a dense urban settlement with high traffic. Since the number of vehicles is more than the number of available SeNBs, the SeNBs can get overloaded with service requests. Vehicles can access the MEC servers through wireless access technologies, and the vehicles acting as mobile edge servers (MES) can provide computational resources to meet the requirements of the network. Furthermore, vehicles in the network can locally execute tasks, offload tasks to nearby MESs, or offload to SESs through the SeNB or MeNBs associated with it. We assume that the MEC servers make decisions about the computation offloading policies of the environment.

B. Communication Model

Let $d_v \in \{0, 1\}, \forall v$ be the local task offloading decision of a vehicle v. If a vehicle decides to locally compute its task, $d_v = 1$, else, if the vehicle decides to offload it to a MES or SES, $d_v = 0$. Also, we denote $s_v \in \{0, 1\} \forall v$ and $m_v \in \{0, 1\} \forall v$ as the task offloading decisions to a SES and MES respectively. If a vehicle v decides to offload its task to a SES, then $s_v = 1$, else $s_v = 0$. If the vehicle offloads the task to a MES $m_v = 1$, otherwise $m_v = 0$. Therefore, the offloading decision profile of the vehicles can be denoted by $d = \{d_v\}_{v \in \mathcal{V}}$, $s = \{s_v\}_{v \in \mathcal{V}}$, and $m = \{m_v\}_{v \in \mathcal{V}}$.

For vehicle to MES communication, if vehicles served by a MES occupy the same frequency spectrum, there will be interference caused by different vehicles. According to the Shannon limit, the spectrum efficiency achieved for the communication link between an MES m and vehicle v is [10]:

$$e_{v,m}^{MES} = log_2(1 + \Omega_{v,m}) \tag{1}$$

where $\Omega_{v,m}$ is the signal-to-interference and noise ratio (SINR) in the communication channel between vehicle, v and MES, m, and is given by:

$$\Omega_{v,m} = \left[\frac{p_v h_{v,m}}{\sum_{j=1}^{J} \sum_{k=1, k \neq m}^{M} p_j h_{j,m} + N_0} \right], \forall v, m \tag{2}$$

where p_v and p_j are the transmission power of the offloading vehicles v and j respectively, $h_{v,m}$, $h_{j,m}$ are the channel gain between vehicle v and MES m, and the channel gain between vehicle j and MES m respectively. J is the total number of interfering vehicles, M is the total number of interfering MESs, N_0 is the power spectrum density of additive white Gaussian noise, and the expression $\sum_{j=1}^{J} \sum_{k=1, k \neq m}^{M} p_j h_{j,m}$ represents the interference from other vehicles.

If $f_{v,m}$ is the fraction of spectrum allocated to vehicle v by the *m-th* MES and B_0 is the available spectrum. Then, the transmission rate of vehicle v is expressed as:

$$R_{v,m}^{MES} = d_v m_v f_{v,m} B_0 e_{v,m}^{MES} \tag{3}$$

In the network, we assume that each offloading vehicle is orthogonally assigned a spectrum from the SeNBs, and there is no interference from vehicles within a single SeNB. However, there is interference from a neighboring SeNB. Therefore, the spectrum efficiency is given by:

$$e_{v,s}^{SES} = log_2(1 + \Omega_{v,s}) \tag{4}$$

where $\Omega_{v,s}$ is the SINR in the communication channel between the vehicle, V, and SES, s, and is given by:

$$\Omega_{v,s} = \left[\frac{p_v h_{v,s}}{\sum_{j=1}^{J} \sum_{k=1, k \neq s}^{S} p_j h_{j,s} + N_0} \right], \forall v, s \tag{5}$$

where p_v and p_j denote the transmission power of the offloading vehicles v and j respectively, $h_{v,s}$, $h_{j,s}$ are the channel gain between vehicle, v and SES, s, and the channel gain between vehicle j and SES s respectively. J is the total number of interfering vehicles, S is the total number of interfering SESs, N_0 is the power spectrum density of additive white Gaussian noise, and the expression $\sum_{j=1}^{J} \sum_{k=1, k \neq s}^{S} p_j h_{j,s}$ is the interference from vehicles from a neighboring SeNB.

Let $f_{v,s}$ be the fraction of spectrum allocated to vehicle v by the *s-th* SES and B_1 be the available spectrum. The transmission rate of vehicle v can be calculated as:

$$R_{v,S}^{SES} = d_v s_v f_{v,s} B_1 e_{v,s}^{SES} \tag{6}$$

C. Computation Model

The computation tasks, A_v of each vehicle is denoted as $A_v := (W_v, C_v, D_v)$. Where W_v denotes the size of the task requiring computation, C_v is the number of CPU cycles required to complete the computation of the task, and D_v is the task's maximum delay. It is possible for a vehicle to process its task locally by utilizing its computing resources or offload its task to nearby edge servers, taking advantage of the server's computing resources. Hence, the computation delay depends on the offloading decision profiles of the vehicles. In the following, we discuss the three cases of task computation, including local, MES, and SES computing.

1) Local Computing: Here, we consider that vehicles have different computation capabilities and have computation resources denoted by U_v. If a vehicle decides to process its tasks, A_v locally, the total task execution time is given by: $T_v = C_v / U_v$.

2) MES Computing: If a vehicle chooses to transfer its task to a nearby MES, the offloading vehicle wirelessly transmits the task to the MES. The total cost of executing the offloaded task depends on the communication cost incurred from the transmission and the cost incurred from the computation on the MES. Let $U_{v,m}$ be the total available computing resources on the MES, w be the fraction of the task to be offloaded, and $F_{v,m}$ be the fraction of the resources allocated for computing the task A_v. Since the task size is W_v and the transmission rate of offloading to a MES is $R_{v,m}^{MES}$, the offloading time cost can be expressed as:

$$T_{v,m}^{MES} = \frac{wW_v}{R_{v,m}^{MES}} \tag{7}$$

and the time for computing the task A_v on the MES is:

$$T_m = \frac{wC_v}{F_{v,m} U_{v,m}} \tag{8}$$

Thus, total task offloading and computing time is given as:

$$T_v^{MES} = T_{v,m}^{MES} + T_m \tag{9}$$

3) SES Computing: If a vehicle offloads to a SES, the offloaded task is transmitted through wireless communication links to the SeNB and then transferred to the MEC server via wired communication links. We neglect the transmission time between the SeNB and the MEC server since the time for the wired link transmission is relatively negligible. When a vehicle offloads a task to the SES, it incurs costs due to offloading to the SES and the processing costs incurred by processing with the SES's resources. Let $U_{v,s}$ be the total available computing resources on the SES, and $F_{v,s}$ be the fraction of the resources allocated for computing the task A_v. Since the size of the task is W_v and the transmission rate of offloading to a SES is $R_{v,s}^{SES}$, the offloading time cost is given as:

$$T_{v,s}^{SES} = \frac{wW_v}{R_{v,s}^{SES}} \tag{10}$$

and the time for computing the task A_v on the SES is:

$$T_s = \frac{wC_v}{F_{v,s}U_{v,s}} \tag{11}$$

Thus, total task offloading and computing time is given by:

$$T_v^{SES} = T_{v,s}^{SES} + T_s \tag{12}$$

D. Energy Model

The energy consumption in the system consists of the local energy consumption in each vehicle and the energy consumed when tasks are offloaded and executed on the MES or SES.

1) Local Energy Consumption: If clock frequency available in each vehicle is U_v, and k is the effective switched capacitance depending on the architecture of the CPU available in the vehicle [11]. The energy consumption required to execute task A_v can be expressed as: $E_v = kC_vU_v^2$.

2) MES Energy Consumption: For offloading to MES, the total energy consumption consists of the energy required to offload the task and the energy consumed during the actual task computation. Let $U_{v,m}$ be the clock frequency available in each MES for task computation, the total energy consumption for tasks offloaded to MES can be expressed as:

$$E_{v,m} = p_v \frac{wW_v}{R_{v,m}^{MES}} + kwC_vU_{v,m}^2 \tag{13}$$

3) SES Energy Consumption: Similarly, for tasks processed by SES, the total energy consumption consists of the energy required to offload to the SES and the energy consumed during the actual task processing. Let $U_{v,s}$ be the clock frequency available in each SES for task computation, the total energy consumption for tasks offloaded to SES can be expressed as:

$$E_{v,s} = p_v \frac{wW_v}{R_{v,s}^{SES}} + kwC_vU_{v,s}^2 \tag{14}$$

IV. Problem Formulation

In the vehicular MEC environment, vehicles send details of their tasks, the available computational capabilities, and energy resources to the MEC servers. The system utilizes the information to manage the spectrum and provide resources for computation. Thus, vehicles can offload their tasks to MEC servers for processing and receive the results when completed. If a MEC server cannot conclude a job within the specified time delay D_v, it will notify the vehicle of failure, reinitiating a new computation process. To satisfy the QoS requirements of the system, the task offloading problem is formulated to minimize the delay and energy consumption costs of the system.

A. System Cost Function

The overall system cost consists of the delay and energy consumption costs due to local computing and offloading decisions made by the vehicles. The delay cost for a vehicle, v at time, t, can be expressed as:

$$T_{total_v}(t) = d_vT_v(t) + T_{v,m}^{MES}(t) + T_{v,s}^{SES}(t) \tag{15}$$

and the energy cost is given by:

$$E_{total_v}(t) = d_vE_v(t) + E_{v,m}(t) + E_{v,s}(t) \tag{16}$$

Therefore, the overall system cost can be expressed as:

$$C(t) = \sum_{v=1}^{V} T_{total_v}(t) + \sum_{v=1}^{V} E_{total_v}(t), \forall v \in V \tag{17}$$

where $T_{total_v}(t)$ is the delay cost associated with vehicle, v at time, t, and $E_{total_v}(t)$ is the energy cost associated with vehicle, v at time, t.

B. Cost Minimization Problem Formulation

The main objective of the system is to minimize the overall cost of the system with a offloading policy that ensures an optimal trade-off between task computation delay and energy consumption. Thus, the optimization problem is given as:

$$\begin{aligned}
\min_{d_v,s_v,m_v,F_{v,s},F_{v,m}} \quad & \sum_{t=1}^{T} C(t) \\
\text{s.t.} \quad C1: \quad & \sum_{v\in V} d_v + s_v + m_v = 1, \quad \forall v \\
C2: \quad & R_{v,s}(t) \leq R_{v,s}, \quad \forall t \\
& R_{v,m}(t) \leq R_{v,m}, \quad \forall t \\
C3: \quad & \sum_{j=1}^{J} \sum_{k=1,k\neq m}^{M} p_jh_{j,m} \leq O_{v,m} \\
C4: \quad & \sum_{j=1}^{J} \sum_{k=1,k\neq s}^{S} p_jh_{j,s} \leq O_{v,s} \\
C5: \quad & T_{total}(t) \leq D_v, \quad \forall t \\
C6: \quad & E_{total}(t) \leq E_{max}, \quad \forall t
\end{aligned} \tag{18}$$

C1 - C6 denotes the constraints of the optimization problem where C1 guarantees that a vehicle locally executes its task, or offloads the task to either a MES or SES for computation. C2 ensures that the total available spectrum is more than the spectrum used by the vehicles during task offloading. C3 and C4 guarantee an acceptable data rate, ensuring that the interference from other vehicles on the MES and SES do not exceed a predefined threshold, $O_{v,m}$ and $O_{v,s}$, respectively. C5 guarantees that a task's total execution time is less than the maximum delay allowed for that task. C6 ensures that the energy consumption due to task offloading and computation is less than the maximum energy, E_{max} of the system.

The solution to this problem gives the outcome for the decision variables d_v, s_v, and m_v. However, these variables are binary, and therefore, the problem is non-convex. Moreover, the system considered is characterized by a dynamic network and heterogeneous computation and energy capabilities, making the optimization difficult to solve. To solve the problem, we propose a DRL-based approach.

V. Reinforcement Learning-Based Solution

Obtaining an optimal solution for computation offloading in the vehicular network is crucial to ensuring that delay-sensitive tasks can meet their deadlines while optimizing energy consumption. Hence, we model the original optimization problem into an MDP and use DRL-based approach to solve it. To successfully formulate the network as an MDP, we identify the state and action space, reward function, and the transition probability of the system. The MDP is expressed as a 4-tuple $(\mathcal{X}, \mathcal{A}, \mathcal{R}, \mathcal{P})$, where $\mathcal{X}$ is the set of possible states of the environment, $\mathcal{A}$ is the set of possible actions for each state, $\mathcal{R}$ is the reward for each state and action pair, and $\mathcal{P}$ is the probabilities of moving from one state to another.

We introduce the DRL architecture consisting of the state space, agent space, reward, and transition probability functions of the system, and then present the TD3 algorithm for solving the cost minimization problem.

A. State, Action, and Reward Definition

The DRL-based solution is characterized by the state of the environment, the action taken by an agent, the reward maximized by the agent, and the transition probability of the environment. Following is the description of these concepts:

1) Environment state: The MEC server receives updates about the state and the task information of an offloading vehicle at time, t. The agent then observes the network and collects information about the system. If there are V vehicles under a service area at time, t, the state of the system, $x(t) \in X$ can be expressed as:

$$x(t) = [R_V^{MES}, m(t), R_V^{SES}, m(t), W_V, C_V, C_{V,m}(t), C_{V,s}(t), \\ E_V(t), E_{V,m}(t), E_{V,s}(t), V(t)] \tag{19}$$

2) Action space: The agent will make decisions based on the observations of the environment state. The agent's action determines the offloading strategy that guides if local, MES, or SES computing is selected. Also, based on the offloading scheme, the agent will decide the appropriate percentage of resources required to complete the tasks at the MEC servers. Hence, the action space, $a(t)$, can be expressed as:

$$a(t) = [d_V, s_V, m_V, F_{V,m}(t), F_{V,s}(t)] \tag{20}$$

3) Reward Function: After taking an action, $a(t)$, the agent receives a reward, $r(t)$, such that the reward is an effect of the previous action. To guarantee that the network QoS demands is satisfied, the delay and energy requirements must also be satisfied. Hence, we define the reward function as:

$$r(t) = -\left[\sum_{v \in V} T_{total_v}(t) + \sum_{v \in V} E_{total_v}(t)\right] \tag{21}$$

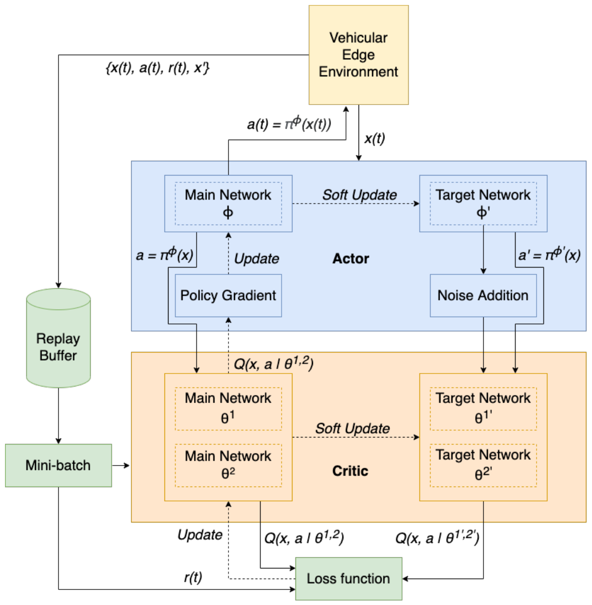

Fig. 2: Architecture of the TD3-based solution

4) Transition Probability Function: The next state x' in MDP does not depend on the previous state, but only on the current state, x. The transition probability function shows the probability of passing from x to x' and can be expressed as:

$$p(x'|x, a) = Pr[x(t+1) = x'|x(t) = x, a(t) = a] \tag{22}$$

where x, a is the state-action pair at time step, t. The current state and action of the system at time, t are denoted by $x(t)$ and $a(t)$, respectively. The state of the environment at time step $(t+1)$ is denoted by $x(t+1)$.

B. TD3 Algorithm

In the vehicular edge environment, moving vehicles in each SeNB compute tasks locally or offload to a SES or a MES in each time slot based on the agent's decision. Also, the agent determines the resources to make available for the computation of each task. Since vehicles are mobile nodes, the available resources and the transmission channel state of the network is dynamic, making decisions based on current observation alone difficult. To overcome these challenges, we use a model-free method based on TD3 for the decision-making [12].

Fig. 2 shows the architecture of the TD3-based algorithm. For each SeNB, the state space is given by eq. 19. At each time step, t, the agent selects an action, a based on offloading policy, π. For every action, the agent receives a reward, r and observes the new state, x'. The discounted sum of rewards received by the agent gives the return, G that the agent accumulates and can be expressed as:

$$G = \left[\sum_{b=0}^{\infty} \gamma^b r(t+b) | \pi, x(t) = x, a(t) = a\right], \gamma \in [0, 1] \tag{23}$$

where γ is the discount factor that determines the weight of present and future rewards. The DRL model goal is to find the optimal offloading policy π^* that maximizes the long-term expected return, $\mathbb{E}\{Q^{\pi}(x,a)\}$. $Q^{\pi}(x,a)$ gives the reward value of each state-action pair and can be expressed as:

Algorithm 1 TD3-Based Solution

```
1: INITIALIZE replay buffer $\mathcal{Z}$
2: INITIALIZE main and target actor networks with weights $\phi$ and $\phi'$, respectively.
3: INITIALIZE main critic networks with weights $Q(x,a|\theta^1)$ and $Q(x,a|\theta^2)$, respectively
4: INITIALIZE target critic networks with weights $Q(x,a|\theta^{1'})$ and $Q(x,a|\theta^{2'})$, respectively
5: for i=1, ..., Number Of Episodes do
6:     INITIALIZE the observation state $x(t) = x(0)$
7:     while time step $t < T$ do
8:         Receive observation from the vehicular environment and collect state $x(t)$
9:         Each vehicle select action $a(t)$ and decide whether to compute task locally, or send task to either a SES or MES
10:        Compute the reward $r(t)$ and estimate the next state $x(t+1)$
11:        if $experiences < Z$ then
12:            Store $(x(t), a(t), r(t), x'(t))$ in $\mathcal{Z}$ for all vehicles $v \in V$
13:        else
14:            Replace first experience with $(x(t), a(t), r(t), x'(t))$ in $\mathcal{Z}$ for all vehicles $v \in V$
15:            Sample mini-batch of $(x, a, r, x')$ for all vehicles from $\mathcal{Z}$
16:            Evaluate $y \leftarrow r(t) + \gamma \min_{i=1,2} Q(x', \tilde{a}|\theta^{i'}), i = 1, 2$
17:            Update the weights $\theta^1$ and $\theta^2$ of the main critic networks using,
18:            $L(\theta^i) \leftarrow \frac{1}{Z}\sum_{j=1}^{M}[y - Q(x,a|\theta^{i'})]^2, i = 1, 2$
19:            if $t \mod d$ then
20:                Update the weights $\phi$ of the main actor network using,
21:                $\Delta_\phi J(\phi) = \frac{1}{Z}\sum\left[\Delta_\phi Q(x,a|\theta^1)|_{a=\pi^\phi(x)}\Delta_\phi \pi^\phi(x)\right]$
22:                Update the target networks using,
23:                $\phi' \longleftarrow \tau\phi + (1-\tau)\phi'$
24:                $\theta^{i'} \longleftarrow \tau\theta^i + (1-\tau)\theta^{i'}, i = 1, 2$
25:            end if
26:        end if
27:    end while
28: end for
```

$$Q^{\pi}(x,a) = \mathbb{E}\left[\sum_{b=0}^{\infty}\gamma^b r(t+b)|\pi, x(t)=x, a(t)=a\right], \gamma \in [0,1] \quad (24)$$

The TD3-based solution consists of an actor network with parameter ϕ and two critic networks with parameters θ^1 and θ^2 for the main network. The solution also has a target network which consists of an actor network with parameter ϕ' and two critic networks with parameters $\theta^{1'}$ and $\theta^{2'}$. In addition, the solution uses a replay buffer to store the experiences from the environment that are used to train the actor and critic networks.

Algorithm 1 describes the architecture of the proposed approach. We start in line 1-4 by randomly initializing the parameters of the neural networks and set the capacity of the replay buffer, $\mathcal{Z}$ to 10^5. Based on the actor network, $\pi^{\phi}(x)$, the agent selects an action in each training episode with some exploration noise ϵ in line 5-9. The vehicles then decide on whether to compute tasks locally or offload to an available MES or SES based on the chosen action. Also, each MEC server decides on the fraction of resources to make available for each offloaded task. After executing the action, the agent will receive an immediate reward, $r(t)$ and observe the new state, x' which are stored in the replay buffer, $\mathcal{Z}$, as a tuple,$(x(t), a(t), r(t), x')$ in line 10-14.

To prevent over-fitting, we add a random noise, $\tilde{\epsilon}$ to the action using:

$$\tilde{a}(x') = a(x') + clip(\epsilon - c, c), \quad \epsilon \sim N(0,\sigma) \quad (25)$$

where $a(x')$ is the action taken with respect to the next state, x_{t+1} and σ is the standard deviation that defines the noise policy. Then, we evaluate the target value, y in line 16, using:

$$y = r(t) + \gamma \min_{i=1,2} Q(x', \tilde{a}|\theta^{i'}), i = 1, 2 \quad (26)$$

where $0 < \gamma < 1$. In line 17-18, the minimum values from the two Q-functions, $Q(x,a|\theta^{1'})$ and $Q(x,a|\theta^{2'})$ are obtained by minimizing the loss functions of the critic given as:

$$L(\theta^i) = \frac{1}{Z}\sum_{j=1}^{Z}[y - Q(x,a|\theta^{i'})]^2, i = 1, 2 \quad (27)$$

The weight, ϕ of the main actor network is updated in line 20-21 after every d time steps using the deterministic policy gradient algorithm given by [13]:

$$\Delta_\phi J(\phi) = \frac{1}{Z}\sum\left[\Delta_\phi Q(x,a|\theta^1)|_{a=\pi^\phi(x)}\Delta_\phi \pi^\phi(x)\right] \quad (28)$$

where Z is the mini-batch of transitions from the replay buffer. In line 22-24, we update the target networks every d time steps using:

$$\phi' \longleftarrow \tau\phi + (1-\tau)\phi' \quad (29)$$

$$\theta^{i'} \longleftarrow \tau\theta^i + (1-\tau)\theta^{i'}, i = 1, 2 \quad (30)$$

where τ represents the rate of updating the network.

VI. Simulation Results and Analysis

This section presents the simulation results to demonstrate the performance of our proposed approach. We first learn the model for the vehicular edge environment and then test the model under different network conditions. Here, we consider a MeNB in $150 \times 150 m^2$ area and 20 SeNBs, with each SeNB associated with an SES server. There are 10 MES and $4 - 10$ offloading vehicles connected to each SeNB at any time. Other parameters used for the simulation are summarized in Table I.

We demonstrate the performance of proposed approach for task offloading in MEC environment using four baselines: *Stochastic*, *Greedy*, *DDPG with MES and SES*, and *TD3 without MES*.

We analyze the TD3-based solution using different learning rates, γ. The simulation results in Fig. 3 show that increasing learning rates makes the agent learn faster. However, with large γ values, the agent may not reach the global solution by converging at a local optimum. Since the results indicate that using γ values of 0.01 and 0.1 leads to a sub-optimal

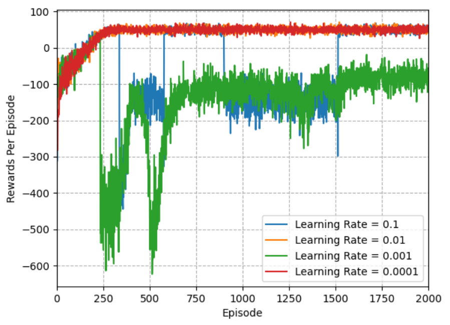

Fig. 3: Convergence performances for different learning rates

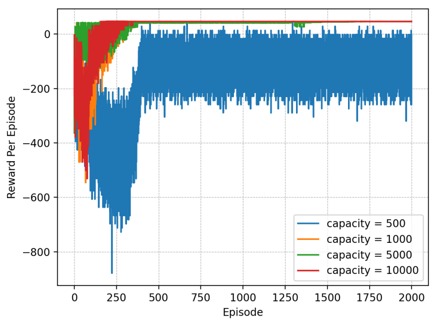

Fig. 4: Convergence performance for different buffer capacity

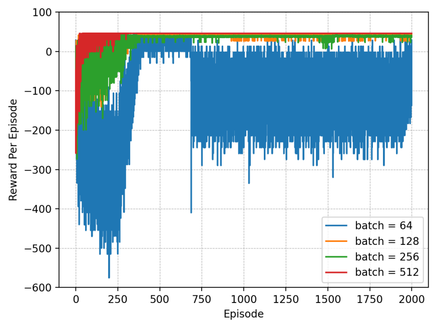

Fig. 5: Convergence performance for different batch sizes

TABLE I: Simulation Parameters

Parameter	Value	Parameter	Value
B_0	5 MHz	C_v	2000 Megacycles
B_1	10 MHz	k	10^{-11}
p_v	0.1 W	$U_{v,s}$	150 GHz
N_0	-100 dBm	$U_{v,m}$	20 GHz
U_v	20 GHz	W_v	$[0.2, 1.2]$ Kbits
Discount Factor	0.99	Replay Memory Size	10^5
Learning rate	10^{-3}	Mini-batch	128

solution, we chose a γ value of 0.0001 for the agent to learn more optimal weights and settle on a global solution. With this learning rate, the agent converges after 250 training episodes, showing that the proposed approach can find the offloading strategy required to minimize system cost. As described in Algorithm 1, the actor and critic networks and the replay buffer have weights requiring initialization. Fig. 4 shows the convergence with different buffer sizes from 500 to 10,000. We find that increasing the buffer size allows the algorithm to converge better. We use a replay buffer of 10,000, meaning that once the algorithm has saved 10,000 experiences, the agent samples mini-batches of experiences to the update actor and critic networks. This size allows the agent to learn from a large pool of experience that may be difficult to save with a small batch size. Fig. 5 shows that with a small batch size of 64, the agent finds it difficult to efficiently utilize the experiences stored in the buffer. With larger batch sizes, the reward increases since the agent can sample more efficiently from the experiences. However, large batch sizes increase training time. Therefore, we choose a batch size of 128 for our experiment. The weights of the actor and critic networks are initialized by TensorFlow, an open-source library used for machine learning. We set the number of episodes to 2000 and the number of steps in an episode to 1000.

For an efficient network, the offloading scheme must make optimal decisions under different network scenarios, such as heterogeneous computing resources or an increased number of offloading vehicles in the network. Fig. 6 illustrates the effect of dynamic computing resources available at the SESs on system cost under different scenarios. For this experiment, we vary the total computing resources available in all SESs and record the system cost, averaging over 3,000 environment states. As shown, using proposed TD3-based solution with cooperative MESs offers the lowest system cost compared to other approaches. Specifically, the system cost of our algorithm is 40.10%, 81.82%, 1.57%, and 34.06% lower than the stochastic, greedy, DDPG with MES and SES, and the TD3-based solution without MES, respectively. Overall, the system cost for each scenario reduces with increasing computing resources on the SES as more resources imply faster computation time for each task. The result illustrates that even with low available computing resources at the SES, the proposed approach minimizes the system cost in the vehicular edge environment and can therefore guarantee acceptable QoS.

We expect that the size of tasks in the network will be dynamic and that the vehicles with extensive tasks can arrive in the network. As the size of the task requiring computation increases, we also expect an increase in the offloading and processing time of the data in the infrastructure. This is because larger task sizes will take longer to be offloaded to either MES or SES. In addition, these tasks will also need additional time for computation and may create a computation bottleneck that increases system latency. To minimize the system's cost, the agent will adjust available resources for each task and determine the decision profile of each vehicle. The agent should still be able to optimize and achieve a minimal delay in the system regardless of the size of the task requiring computation. Fig. 7 demonstrates the total network delay with different task sizes requiring computation. As we expect, the delay in the system increases as the size of the task increases. However, the TD3-based approach with cooperative MESs still achieves the lowest delay compared to the other approaches. Our approach achieves an average of 4.54%, 63.75%, 1.23%, and 34.21% delay lower than the stochastic, greedy, DDPG with MES and SES, and the TD3-based solution without MES, respectively. This result shows that tasks can still be computed in the fastest possible duration while optimizing the energy usage in the network.

Fig. 8 shows the energy use in the network with a different maximum number of vehicles in each SES. There can be varying task size in each SES's coverage area, and there can be a different number of offloading vehicles in the SES below the maximum value at any given time step. From the results, only a slight increase in energy use is observed with an increase in the number of vehicles. However, the proposed TD3-based solution with cooperative MESs still shows that the

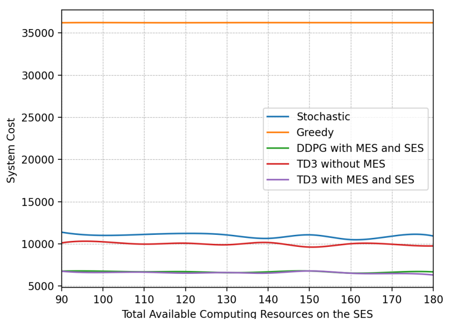

Fig. 6: Cost with available SES computing resources

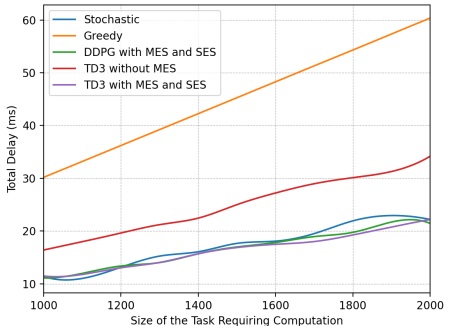

Fig. 7: Delay with different task size for vehicles in each SES

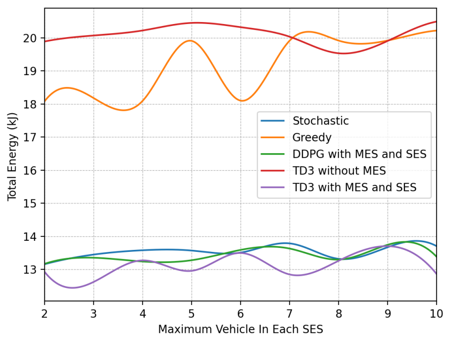

Fig. 8: Energy with maximum number of vehicles in each SES

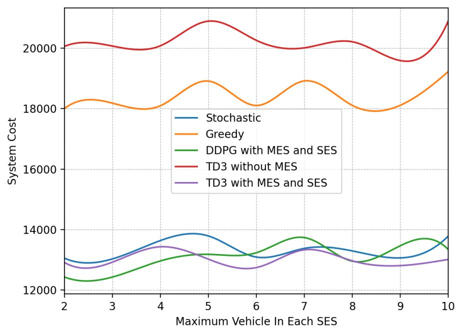

Fig. 9: Cost with maximum number of vehicles in each SES

energy usage in the system is minimized more efficiently when compared with the baseline solutions. Using our approach, the infrastructure consumes 3.12%, 31.58%, 2.27%, and 34.83% less energy than the stochastic, greedy, DDPG with MES and SES, and the TD3-based solution without MES, respectively. This result illustrates that the proposed solution can optimize energy usage even with additional network load and can minimize the overall delay of the network.

In Fig. 9, we also illustrate the effect of increasing the maximum number of vehicles in each SES has on the system cost. Initially, the DDPG-based scheme seems to perform better with less than five vehicles in each SES. However, as the network becomes more complex with increasing loads, the proposed TD3-based solution begins to show better performance. With 10 vehicles in each SES, our approach outperforms the stochastic, greedy, DDPG with MES and SES, and the TD3-based baseline solution without MES by 5.58%, 32.29%, 2.48%, and 37.71%, respectively. Since TD3 does well in complex scenarios and can optimally manage the available system resources, the agent follows an offloading policy that minimizes the overall system cost under different network conditions.

VII. Conclusion

In this paper, we present a task offloading scheme aimed at enhancing system delay and energy consumption in vehicular edge networks. The proposed solution leverages vehicles in the network as cooperative MESs to support MEC servers, thus reducing the requests load on SESs and minimizing the system cost. This cooperative decision-making by vehicles, SESs, and MESs optimizes network delay and energy consumption. Also, the approach ensures that vehicles can offload tasks to available SES or MES in their coverage areas, and the SES and MES can decide on the percentage of resources to allocate for each offloaded task, thereby reducing computation costs. To address the complexity of the problem, we model the offloading problem as an MDP and employ a TD3-based DRL algorithm to solve it. Our results demonstrate the superiority of the proposed method compared to DDPG-based or TD3-based baseline solutions without cooperative MESs.

References

[1] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, “Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 1484–1494, 2020.

[2] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, “Survey on multi-access edge computing for internet of things realization,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, 2018.

[3] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, “Reinforcement learning for resource provisioning in the vehicular cloud,” *IEEE Wireless Communications*, vol. 23, no. 4, pp. 128–135, 2016.

[4] R. Q. Hu *et al.*, “Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning,” *IEEE Transactions on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, 2018.

[5] Y. Liu, H. Yu, S. Xie, and Y. Zhang, “Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks,” *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, pp. 11 158–11 168, 2019.

[6] H. Peng and X. Shen, “Deep reinforcement learning based resource management for multi-access edge computing in vehicular networks,” *IEEE Transactions on Network Science and Engineering*, vol. 7, 2020.

[7] B. Hazarika, K. Singh, S. Biswas, and C.-P. Li, “Drl-based resource allocation for computation offloading in iov networks,” *IEEE Transactions on Industrial Informatics*, 2022.

[8] “Twin delayed ddpg,” *Open AI, Spinning Up*. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/td3.html

[9] A. Mukherjee, “Macro-small cell grouping in dual connectivity lte-b networks with non-ideal backhaul,” in *2014 IEEE International Conference on Communications (ICC)*. IEEE, 2014, pp. 2520–2525.

[10] C. E. Shannon, “Communication in the presence of noise,” *Proceedings of the IRE*, vol. 37, no. 1, pp. 10–21, 1949.

[11] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, “Energy-optimal mobile cloud computing under stochastic wireless channel,” *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, 2013.

[12] S. Fujimoto, H. Hoof, and D. Meger, “Addressing function approximation error in actor-critic methods,” in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[13] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in *International conference on machine learning*. PMLR, 2014, pp. 387–395.