Journal of King Saud University – Computer and Information Sciences 35 (2023) 101512

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

DRL-assisted delay optimized task offloading in automotive-industry 5.0 based VECNs

Muhammad Ayzed Mirza [a], Junsheng Yu [a,b,c,*], Salman Raza [d], Moez Krichen [e], Manzoor Ahmed [f], Wali Ullah Khan [g], Khaled Rabie [h,i,*], Thokozani Shongwe [i]

[a] BUPT-QMUL EM Theory and Application International Research Lab, Beijing University of Posts and Telecommunications, Beijing 100876, China
[b] School of Physics and Electronic Information, Anhui Normal University, Wuhu 241003, China
[c] School of Intelligence and Digital Engineering, Luoyang Vocational College of Science and Technology, Luoyang 471000, China
[d] Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan
[e] Faculty of CSIT, Al-Baha University, Saudi Arabia ReDCAD Laboratory, University of Sfax, Tunisia
[f] School of Computer and Information Science and also with Institute for AI Industrial Technology Research, Hubei Engineering University, Xiaogan 432000, China
[g] Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, Luxembourg City 1855, Luxembourg
[h] Department of Engineering, Manchester Metropolitan University, Manchester M15 6BH, UK
[i] Department of Electrical and Electronic Engineering Technology, University of Johannesburg, Auckland Park, Johannesburg 2006, South Africa

ARTICLE INFO

Article history:
Received 19 January 2023
Revised 10 February 2023
Accepted 11 February 2023
Available online 24 February 2023

Keywords:
Automotive-Industry 5.0
Vehicular Edge Computing (VEC)
Task offloading
Beyond fifth-generation (B5G)
Deep Reinforcement Learning (DRL)

ABSTRACT

The rapid growth of Automotive-Industry 5.0 and its emergence with beyond fifth-generation (B5G) communications, is making vehicular edge computing networks (VECNs) increasingly complex. The latency constraints of modern automotive applications make it difficult to run complex applications on vehicle on-board units (OBUs). While multi-access edge computing (MEC) can facilitate task offloading to execute these applications, it is still a challenge to access them promptly and optimally. Traditional algorithms struggle to guarantee accuracy in such dynamic environment, but deep reinforcement learning (DRL) methods offer improved accuracy, robustness, and real-time decision-making capabilities. In this paper, we propose a DRL-based mobility, contact, and load aware cooperative task offloading (DCTO) scheme. DCTO is designed for both cellular and mmWave radio access technologies (RATs), and both binary and partial offloading mechanisms. DCTO targets delay minimization by opportunistically switching RATs and offloading mechanisms. We consider relative efficacy and neutrality factors as key performance indicators and use them to derive the DRL agent's reward function. Extensive evaluations demonstrate that the DCTO scheme exhibits a substantial enhancement in task success rate, with an increase from 2.61% to 21.34%. It also improves the efficacy factor from 1.38 to 3.52 and reduces the neutrality factor from 4.99 to 0.76. Furthermore, the average task processing time is reduced by a range of 3.77% to 24.15%. Additionally, the DCTO scheme outperforms the other evaluated schemes in terms of reward and TFPS ratio.

1. Introduction

Recent technological advancements in the Automotive-Industry 5.0 and smart vehicles along with the emergence of beyond fifth-generation (B5G) communication, the vehicular edge computing networks (VECNs) are becoming more convoluted (Khan et al., 2022). The goal of Automotive-Industry 5.0 is to enable the intelligent linkage among humans and connected and autonomous vehicles (CAV) utilizing cutting-edge technologies like 6G communications, machine learning tools, edge computing, and more (Ahmed et al., 2022; Liu et al., 2022). The real-time intelligence brought by the smart vehicles greatly uplifted the interconnectivity, sharing dynamic road safety data, and quality of

* Corresponding authors.
E-mail addresses: mamirza@bupt.edu.cn (M.A. Mirza), jsyu@bupt.edu.cn (J. Yu), salmanraza@ntu.edu.pk (S. Raza), moez.krichen@redcad.org (M. Krichen), waliullah.khan@uni.lu (W.U. Khan), k.rabie@mmu.ac.uk (K. Rabie), tshongwe@uj.ac.za (T. Shongwe).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.02.013
1319-1578/

experience (QoE) in VECNs (Tang et al., 2021). However, vehicular intelligent and immersive applications are also increasing swiftly to accompany the QoE but with the increased resources requirements and latency sensitivity. On the other hand, the computation power of modern vehicles alone, is not enough to meet the requirements of these applications (Jiang et al., 2021).

To overcome this challenge, the concept of computation/task offloading has been widely employed, and traditionally done through multi-access edge computing (MEC), and vehicular edge computing (VEC). MEC, and VEC being the edge computing paradigms, are installed in closer proximity to the vehicles at serving evolved-node-B (eNB), road side units (RSUs), or at parking lots. Therefore, are effective in reducing transmission and processing delay and improving energy consumption. Besides, Vehicle-vehicle, vehicle-to-edge, and edge-to-edge both horizontal and vertical collaboration is also widely proposed in research to addresses these latency and load unbalancing issues (Ahmed et al., 2022).

In addition, to perform the task offloading operation, various radio access technologies (RATs) options are available. For instance, dedicated short-range communications (DSRC) and cellular (i.e., 4G/5G/LTE and 5G new radio (NR)). Task offloading is supported by all these RATs, but each of them has different levels of support for the various vehicle to everything (V2X) use cases (Naik et al., 2019). Consequently, their support for high throughput, ultra-low latency, high reliability and accuracy, and high mobility can significantly change the next-generation VECNs and their applications (Shibata et al., 2019).

However, nearby vehicles, vehicle cloudlets, and MEC/VEC servers have considerable computational capabilities. In addition, there are various RATs to communicate with these resource-rich nodes (Boukerche and Sotoro, 2020). Still, offloading decision-making is another important point, because completing the tasks under the given limit depends on this decision.

1.1. Motivation & contributions

Reducing the delays, such as processing, queuing, and transmission, is the most important task offloading objective. However, the dynamic and complex conditions in vehicular edge computing environments pose major challenges for task offloading, including network variability, mobility, resource limitations, high computational demands, and latency restrictions. These challenges require innovative solutions to handle the complexities and deliver accurate results in real-time. RL/DRL methods can be used to address these difficulties by learning directly from the environment (Jin et al., 2022). To tackle the challenges of task offloading in the dynamic VECN environment, we propose a DRL-based task offloading scheme. The key features of the proposed DRL based mobility, contact, and load aware cooperative task offloading (DCTO) scheme are as follows:

System Configuration: A VECN dynamics aware 5G-NR-V2X architecture based system model is followed. Hybrid wireless technologies and transmission methods in sub-6 GHz and 28 GHz frequency bands have been considered to minimize the transmission latency. Vehicles generate tasks, process them locally or generate offloading requests to the VEC server at RSU for processing, while being under RSU's coverage area.

Assessment Structure: The dynamic attributes of tasks and their arrival, communication and availability metrics of computational resources are considered in the system model. Characteristics of the vehicle environment, such as local and edge resources availability, and communication parameters are closely monitored and updated whenever they become operational upon the task arrival. Relative efficacy and neutrality factors are also introduced for fair estimation of computation offloading.

Algorithm Design: Initially, a computation offloading and resource allocation optimization objective is formulated, while ensuring that the whole offloading process complies with delays and resource constraints. Later, this objective is transformed into a Markov decision-making process (MDP) based on the description of states and actions in the dynamic VECNs setup. Then, a DRL-driven DCTO scheme is proposed following proximal policy optimization (PPO) to minimize the overall task processing latency.

Numerical Evaluation: Numerical results show that our proposed DCTO scheme achieves significant advantages over other DRL-based task offloading schemes. Like, more reward in less time with less variability, even training takes less time. Furthermore, better task success and failure ratio, higher average relative efficacy and lower neutrality factor.

The rest of the paper is organized as follows. The system model, including the network, communication, and computation models, is presented in Section 3. The problem formulation and the DCTO scheme are discussed in Section 4. Numerical results and discussions are provided in Section 5, and the paper is concluded in Section 6.

2. Related work

We investigated different task offloading schemes in VECNs. Several approaches for instance, heuristic, greedy, game theory, convex/non-convex optimization, machine learning (ML), and RL/DRL, have been adopted to address the task offloading challenge. Some techniques have focused on minimizing computation and communication delay, cost, and energy, while some other work has been done on optimal resource allocation, MEC/VEC load balancing, server selection, and QoS stability.

For instance, considering the optimization of task acceptance ratio and execution time, Nguyen et al. (2022) targeted the computational resources of parked vehicles. A meta-heuristic edge genetic algorithm (EdgeGA) is proposed to deal with the time complexity and scalability problems of binary integer problems (BIP) while formulating the task offloading problem. A containerized orchestration framework is built for the task offloading problem on Kubernetes. In an other work Du et al. (2022) aimed to reduce the system cost and latency while exerting nonorthogonal multiple access (NOMA) based VEC model. Two heuristic algorithms are proposed, one for offloading decision optimization and the other for vehicular clustering and resource allocations. However, the proposed solution provides a low-cost and latency-optimal solution but comes at the cost of high complexity.

Optimization and game theoretic approaches are also widely applied in computational offloading research. Similarly, Luo et al. (2021) exerted Pareto optimality and particle swarm optimization (PSO) into vehicular task offloading problem. A multi-objective optimization is provided to decrease the delays and system cost while jointly considering the offloading decision and the resources (communication and computation). In another work, Shu and Li (2022) introduced a quantum PSO algorithm and targeted to reduce the latency and energy consumption while resource allocations. In another work Raza et al. (2021) also provided a game-theoretic task offloading scheme while considering both modes (mmWave and cellular) of 5G new-radio-vehicle-to-everything (NR-V2X) radio access technology (RAT). The proposed scheme aims to reduce overall system delay and energy consumption. To further improve the system, Raza et al. extended their work in Raza et al. (2022), and utilized multi-hop V2V communications, vehicular clouds to optimize task's average response time. Similarly, Deng et al. (2021) proposed a user-centered joint optimization loading scheme for edge computing in the Internet of Things (IoT). The goal is to minimize the weighted cost of time delay,

energy consumption, and price while meeting the advanced personalized needs of users. The optimization problem is modeled as a mixed-integer nonlinear programming problem and solved using a branch-and-bound algorithm based on linear relaxation improvement and a particle swarm optimization algorithm based on 0–1 and weight improvement. Likewise, Chen et al. (2022a) introduced a distributed multi-hop task offloading decision model for improved task execution efficiency in MEC based vehicular network. Initially the candidate vehicles are selected based on k-hop wireless communication range. Later, the task offloading problem is modeled as a generalized allocation model and solved using the greedy algorithm and improved discrete bat algorithm. A new approach to Federated Learning (FL) in VEC is proposed by Liu et al. (2021) to address the challenges of communication overheads and data privacy. The proposed approach, called FedCPF, provides a customized local training strategy, a partial client participation rule to reduce uplink congestion, and a flexible aggregation policy to manage communication overheads.

The literature has proposed a number of task offloading approaches using various techniques, such as heuristics, greedy algorithms, game theory, and convex/non-convex optimization, to tackle the challenge of task offloading. The main objective in task offloading is to minimize delays, which encompass processing, queuing, and transmission time (Nguyen et al., 2022; Du et al., 2022; Luo et al., 2021; Shu and Li, 2022; Raza et al., 2021; Raza et al., 2022; Deng et al., 2021; Chen et al., 2022a).. However, vehicular edge computing environments are characterized by dynamic and complex conditions that pose considerable difficulties for task offloading. These conditions can include, among others, network variability, mobility, resource limitations, high computational demands, and latency restrictions. These dynamic challenges make task offloading in vehicular edge computing environments a complex problem that requires new approaches to handle the intricacies and deliver accurate results in real-time (Jin et al., 2022). To overcome these complexities, one approach is to leverage direct learning from the environment through RL/DRL methods (Jin et al., 2022). DRL offers a model-free approach that is capable of handling complex and high-dimensional state spaces, resulting in improved accuracy, robustness, generalization, and real-time decision-making capabilities, which make it a more suitable alternative compared to conventional algorithms (Jin et al., 2022; Liu et al., 2022).

RL/DRL has dramatically anchored state-of-the-art performance in various vehicular environments, particularly the task offloading (Liu et al., 2022). Considering the dynamic nature of the vehicular environment, Shuai et al. (2021) provided a delay optimization scheme. Initially, the transmission latency is minimized using the optimal flow-based routing algorithm, then a deep Q-learning based task offloading strategy selection scheme is used for adaptive task offloading considering the MEC load states. In another work Cui et al. (2021) proposed a multi-objective RL scheme for task offloading. Firstly K-nearest neighbour approach is used for the selection of offloading layers, then a V2V collaborative algorithm is used for V2V computation sharing scenarios. Later, a Q-learning based communication and computation resource allocation algorithm is used in MEC for further cost, latency, and reliability optimization. Similarly, Yao et al. (2022) also presented a twin delayed deep deterministic policy gradient (TD3) based computation offloading scheme. The offloading problem is initially transformed into Markov decision process (MDP), then TD3 is used to find the optimal offloading strategy given the minimum delay and energy consumption. In one of the works, Zhang et al. (2022a) address the limited energy and computing capacity of internet of things (IoT) nodes by integrating wireless power transmission and mobile edge computing. A DRL-based framework is proposed to learn the near-optimal wireless power transmission duration. Another work by Zhou et al. (2020) proposes a solution for task offloading optimization in an Internet of Health Things (IoHT)-based e-Health paradigm. The task offloading problem is formulated as an adversarial multi-armed bandit problem and a ultra-reliable low latency communications (URLLC)-aware task offloading scheme based on the exponential-weight algorithm for exploration and exploitation (EXP3) algorithm is designed to dynamically balance the URLLC constraints and energy consumption. In another work by Qiao et al. (2020), a distributed trustworthy storage architecture with RL is proposed for intelligent transportation systems (ITS). The architecture features an intelligent storage scheme that stores data dynamically based on trustworthiness and popularity, improved resource scheduling, and storage space allocation. A trapdoor hashing based identity authentication protocol is proposed to secure the transportation network access and a federated trusted evaluation model is used to evaluate the trustworthiness of edge servers and data producers.

Holding a trade-off between computation and communication while observing the local, MEC, and cloud resources, Wang et al. (2022) proposed a quantum-inspired RL (QRL) task offloading scheme. The resource management scheme is initially formulated as a delay optimization problem which is later transformed into an MDP problem, then QRL is applied to find out the optimal resource allocation policy. Quantum parallelism is adopted here with QRL algorithm for the acceleration of convergence and to overcome the dimentionality. In another work, Zhang et al. (2022b) also provided an RL-based task offloading algorithm to minimize task execution and communication delays with throughput optimization while using Sub-6 GHz and 28 GHz frequency bands. The task offloading problem is initially formulated into MDP, then a DRL-based deep Q-network (DQN) algorithm is applied for task offloading and resource allocation to reduce latency. Moreover, a federated RL algorithm is used to minimize transmission overheads. Similarly, Chen et al. (2022b) proposed an asynchronous advantage actor-critic (A3C) based task offloading scheme while considering fairness and efficiency. Firstly, joint offloading and resource-allocation are optimized in accordance with delay constraints. Later, the problem is transformed into MDP and A3C is applied for the optimal offloading decisions.

3. System model

This section proposes a vehicle and edge collaboration architecture considering sub-6 GHz and mmWave communication and transmission modes.

3.1. Network and communication model

We consider an architecture where eNBs are connected to the core cloud while serving next-generation Node B (gNB) type RSUs and RSUs serving 5G-NR capable vehicles, as shown in Fig. 1. Each RSU r_i is connected to and can communicate with the attached VEC server, and an eNB or remote cloud center through a dedicated link. There is a parking access point (P-AP) at the parking lot, which is connected to the parked vehicles and the adjacent gNB type RSUs through a dedicated link. The system architecture is supposed to be on 5G NR-V2X RAT with mmWave and Sub-6 GHz spectrum bands. There are two types of vehicles in our model, resourceful vehicles represented by v_j , and resource demanding vehicles v_k . Both v_j, v_k can communicate with each other through r_i and with the VEC server attached to the r_i through Uu links or mmWave links. Here, $i,j,k \in \mathcal{N}$ and $\mathcal{N} = \{1,2,\ldots,\mathcal{N}\}$, and is a natural number.

We have considered that vehicles are equipped with multiple antennas, as they are supposed to communicate on both

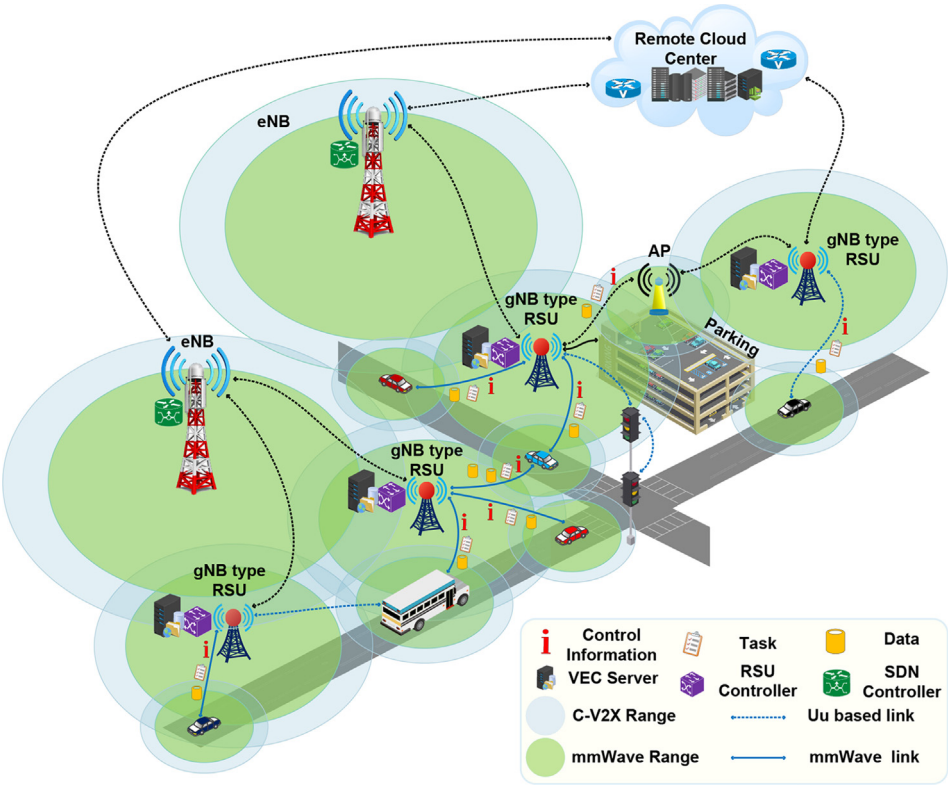


Fig. 1. System model, illustrating the framework mechanism followed to conduct the task offloading under both NR C-V2X and mmWave technologies.

Sub-6 GHz and mmWave links. The allocated spectrum to the v_j and v_k is orthogonal (ITU, 2021). The transmission rate $\Re^{c}_{k,i}$ for cellular link among v_k and the connected r_i is:

$$\Re^{c}_{k,i} = B\log_2\left(1 + \left(p_k d^{-\xi}_{k,i}|h|^2\right)/\sigma^2\right), \tag{1}$$

where B is the channel bandwidth, p_k is vehicle transmission power. $d_{k,i}$ is the distance among v_k and r_i and is defined by $d_{k,i} = [\mathfrak{r}\lceil l_k/\mathfrak{r}\rceil - l_k]$, l_k is the current position of v_k and $\mathfrak{r}$ is the communication footprint radius of r_i. In addition, ξ denotes the path loss exponent (Wang et al., 2016), $|h|^2$ is Rayleigh fading of uplink channel (Wang et al., 2018), and σ^2 is the white Gaussian noise (Raza et al., 2022).

The signal to noise ratio (SNR) of a typical mmWave based link is calculated by following (2) and the data transmission rate is calculated using (3).

$$\begin{aligned} SNR_{k,i} = {} & p^{mm}_{k} - \sigma_{mm} - 10\,log_{10}\,(B_{mm}) \\ & +2\,\mathcal{G}_{r_x,t_x} - 10\,\varsigma\,log_{10}(d_{j,i}) - 69.6 - \rho_{\alpha}, \end{aligned} \tag{2}$$

$$\Re^{mm}_{k,i} = \left(B^{mm}\,log_2\left(1 + SNR_{k,i}\right)\right). \tag{3}$$

P^{mm}_{k} is v_k's transmit power for mmWave, σ^{mm}, B^{mm}, and ς are noise power in the mmWave link, mmWave bandwidth, and the path loss exponent, respectively. $\mathcal{G}_{r_x,t_x}$ is the antenna gain of a generic mmWave based $Rx - Tx$ pair. ρ_{α} is the shadow fading effect modeled as $\rho_{\alpha} = \mathcal{N}(0, \sigma^2)$, a log-normal random variable with zero-mean standard deviation whose value is set to 5.0dB for LOS and 7.6dB for NLOS at 28GHz (3GPP, 2019). Moreover, the contact duration $t_{k,i}$ of v_k with r_i can be calculated as:

$$t_{k,i} = \left(2\sqrt{\mathfrak{r}^2 - \mathfrak{e}^2}\right)/\overrightarrow{\mu_k}. \tag{4}$$

here, $\overrightarrow{\mu_k}$ and $\mathfrak{e}$ are the vector speed of v_k and vertical distance between the road and the r_i, respectively.

3.2. Task model

We consider that the time domain is divided into $\tau_{\mathfrak{n}}$ sections of equi-length $\mathfrak{m}$ t-time slots $(\mathfrak{m}, \mathfrak{n}, t \in \mathcal{N})$. The task generation probability follows a Bernoulli distribution with probability $\mathcal{P}$, hence the mean rate of task arrivals $\lambda_{\vartheta} = \mathcal{P}/t$ (Ye et al., 2021). Each offloading vehicle generates ϑ_k task at the beginning of t-time slot, and each ϑ_k task consists of a 6-tuple $\vartheta_k = \{c_k, s_k, t_k, \varphi_k, x_k, d_k\}$. Here c_k, s_k, and φ_k are the required CPU cycles, task input size in bits, and maximum threshold execution time of task ϑ_k, respectively. Whereas, x_k represents the vehicle v_k's position, and d_k is the distance among serving RSU and v_k. While, t_k is ϑ_k's arrival time.

3.3. Offloading decision model

Each ϑ_k task is associated with a $\mathfrak{d}_k$ offloading decision, where $\mathfrak{d}_k \in \{0, 1\}$. The decision $\mathfrak{d}_k = 0$ corresponds to local processing and $\mathfrak{d}_k = 1$ implies that the task is decided to be offloaded. When ever $\mathfrak{d}_k = 1$, the vehicle also has to decide the optimal offloading decision δ_k. Here, $\delta_k \in \{0, 1\}$, and defines the offloading type, i.e., binary or partial offloading. The δ_k is set to 0 when binary offloading is opted otherwise it is set to 1 for partial offloading.

3.4. Queue, resources state model

Being realistic, v_k can only offload as many ϑ_k tasks or as many portions $\vartheta_{k,z}$ $(z \in \mathcal{N})$ of ϑ_k as it currently has, and follows $s^{o}_{k} \leqslant \mathfrak{q}^{s}_{k} - s^{v}_{k}$, the offloading constraint. All the sizes are in bits. Here $s^{o}_{k}, \mathfrak{q}^{s}_{k}$, and s^{v}_{k} are the offloading task size, queue size, and the local task's size, respectively. The offloading task size s^{o}_{k} also follows $s^{o}_{k} \leqslant \Re_{j,i}$ constraint. The queue size evolution can be expressed as:

$$\mathfrak{q}^{s}_{k}(t+1) = \max\left\{\left(s^{v}_{k} - s^{o}_{k} + \lambda_{\vartheta}\right), 0\right\}, \tag{5}$$

The computation resources (local or edge) allocation can be done only within the one of corresponding t-time slot. The dynamic CPU resource state $\mathfrak{S}_t^{\mathfrak{C}}$ at the starting of t-time slot can be represented as following matrix:

$$\mathfrak{S}_t^{\mathfrak{C}} = \begin{bmatrix} \mathfrak{C}_1, c_1 & \mathfrak{C}_1, c_2 & \dots & \mathfrak{C}_1, c_n \\ \mathfrak{C}_2, c_1 & \mathfrak{C}_2, c_2 & \dots & \mathfrak{C}_2, c_n \\ . & . & \dots & . \\ . & . & \dots & . \\ \mathfrak{C}_t, c_1 & \mathfrak{C}_t, c_2 & \dots & \mathfrak{C}_t, c_n \end{bmatrix} \tag{6}$$

$\mathfrak{C}_t, c_n$, is the residual CPU time of tth time slot for the nth CPU, it ranges from 0 to t.

Since, every task arrives at the start of t-time slot, it may not be processed immediately, as there could be other local and or offloaded tasks in the queue to be processed. Therefore, a task may have to wait for the starting of new t-time slot. Additionally, it has to wait in the in-slot waiting queue before its processing turn. The slot waiting delay can be expressed as $t.\tau_{\mathfrak{n}}$, and the in-slot queuing delay $\mathfrak{t}_q$ can be calculated as in (7).

$$\mathfrak{t}_q = (\tau_{\mathfrak{n}} - \max\{\mathfrak{C}_t, c_1, \mathfrak{C}_t, c_2, \dots, \mathfrak{C}_t, c_n\}). \tag{7}$$

3.5. Computation model

The computing model we followed consists of local computing model and edge computing model, and they are as follows.

1) Local computing model: We assume that all the v_k, the offloading vehicles, have the same CPU frequency, denoted as $\mathfrak{f}_k$. While assuming $\mathfrak{c}$ as CPU cycles required to process one bit of task, then the average local processing delay for task ϑ_k having a size of s_k can be calculated as:

$$\mathfrak{T}_l = \left((1 - \mathfrak{d}_k(1-\delta_k))\left\{t.\tau_{\mathfrak{n}} + \mathfrak{t}_q + \frac{\mathfrak{c}\left(\alpha s_k + s_k^{\mathfrak{v}}\right)}{\mathfrak{f}_k}\right\}, s.t.\ \mathfrak{T}_l \leqslant \varphi_k. \right. \tag{8}$$

here, α is the portion of the task to be processed locally, and has a value range of $0 \leqslant \alpha \leqslant 1$.

2) Edge computing model: Each RSU is considered to be equipped with a multi CPU VEC server, and each VEC server performs parallel computing of various types of tasks. The availability of each CPU will be different at every t-time slot, due to dynamic nature of VECN's. When v_k is unable to process the ϑ_k task within φ_k time, it is decided to be offload at the edge, then $\mathfrak{d}_k$ is set to 1.

The VEC server's CPU resources are assigned to the ϑ_k tasks according to the longest available CPU time in the corresponding t-time slot. Just in case, there are multiple CPUs available with the same CPU time availability, then a random CPU selection is done. Moreover, processing ϑ_k task at VEC also includes the $t.\tau_{\mathfrak{n}}, \mathfrak{t}_q$, and the transmission delays. The waiting time at VEC can be expressed as:

$$\begin{aligned} &\mathfrak{T}_e = \mathfrak{d}_k\left(\frac{\mathfrak{c}\left(\beta s_k^{\mathfrak{v}}\right)}{\mathfrak{f}_e} + t.\tau_{\mathfrak{n}} + \mathfrak{t}_q + \frac{\beta s_k^{\mathfrak{v}}}{\mathfrak{R}_{k,i}}\right), \\ &s.t. \quad \mathfrak{T}_e \leqslant \varphi_k. \end{aligned} \tag{9}$$

here, $\mathfrak{R}_{k,i}$ is the generalized term for transmission rate, either for cellular or mmWave accordingly. When $\mathfrak{T}_e$ complies with the $\mathfrak{T}_e \leqslant \varphi_k, (\beta s_k^{\mathfrak{v}}/\mathfrak{R}_{k,i}) \leqslant t_{k,i}$, and $\mathfrak{d}_k = 1, \text{and} \delta_k = 0$, the value of β becomes 1 and the computation is entirely done by the VEC. The β is the portion of task to be processed at VEC, it have a value range of $0 \leqslant \beta \leqslant 1$. However, both α and β must follow the constraint $\alpha + \beta = 1$.

The decision $\mathfrak{d}_k = 1, \text{and} \delta_k = 0$, is only taken when v_k has no free CPU available to accommodate the ϑ_k at given t-time slot. Otherwise, if v_k has some free CPU available but not enough to complete the ϑ_k task, it must choose partial offloading considering φ_k. In addition, when partial offloading is opted both $\mathfrak{d}_k, \delta_k$ become 1.

4. Problem formulation and the solution

In this section, we formulate the delay minimization problem. We have proposed our solution considering efficacy and neutrality factors for offloading with the aim of reducing the overall task processing delay. The total delay of a given task can be expressed as:

$$\Gamma_k = \{1 - \mathfrak{d}_k(1-\delta_k)\} \times \mathfrak{T}_l + \mathfrak{d}_k \times \mathfrak{T}_e, \tag{10}$$

Γ_k represents the processing delay for single task, the average delay Γ_{avg}, for K total tasks can be determined by following (11).

$$\Gamma_{avg} = \frac{1}{K}\sum_{k=1}^{K}[(1 - \mathfrak{d}_k(1-\delta_k))\mathfrak{T}_l + \mathfrak{d}_k\mathfrak{T}_e]. \tag{11}$$

The lower the value of Γ_{avg}, the more efficient and rationale the allocation of resources will be. Ultimately the system will be able to process more tasks. Therefore, minimizing the average task processing delay is the optimization goal and can be expressed as:

$$\begin{aligned} &\underset{\mathfrak{T}_l,\ \mathfrak{T}_e}{\text{minimize}} \ \frac{1}{K}\sum_{k=1}^{K} \Gamma_{avg}, \\ &s.t.\ C1 : \min\{\mathfrak{T}_l, \mathfrak{T}_e\} \leqslant \varphi_k, \\ &C2 : \forall\ \mathfrak{d}_k \in \{0,1\}, \text{and } \forall\ \delta_k \in \{0,1\}, \\ &C3 : \forall\ k \in \{1,2,3,\dots,K\}. \end{aligned} \tag{12}$$

Here the constraint C1 specifies that the minimum of $\mathfrak{T}_l, \mathfrak{T}_e$ should be less than or equal to the maximum threshold time φ_k to process the ϑ_k task. $\mathfrak{d}_k$ and δ_k are the offloading decision parameters in C2. When the local processing is opted, the decision $\mathfrak{d}_k$ becomes 0. However, when there is no free CPU available locally on device at the given time slot the $\mathfrak{d}_k$ becomes 1, but δ_k remains 0. Moreover, when there is free CPU available locally but is not satisfying the condition $\mathfrak{T}_l \leqslant \varphi_k$, then, δ_k becomes 1 along-with the $\mathfrak{d}_k$. At this moment, partial offloading will be opted, i.e., α part of ϑ_k will be processed locally according to the available CPU, and β part will be offloaded to VEC. In addition, when partial offloading is opted, the total latency Γ_k must follow the φ_k constraint to process ϑ_k.

Due to the dynamics of the VECN environment, the information of the tasks/sub-tasks uploaded to the VEC is not the same for each t-time slot. Consequently, in this type of situation, rational comparison should be followed for evaluation. Therefore, we introduce the efficacy and neutrality factors as follows. The local completion delay Γ_b for a given task ϑ_k is considered as a benchmark to facilitate the comparison.

1) Efficacy factor: The product of difference between Γ_b, Γ_k and φ_k, Γ_k is considered as relative latency saved from processing. This will show the effectiveness of the offloading decision, and can be expressed as:

$$\Gamma_k^s = (\Gamma_b - \Gamma_k) \times (\varphi_k - \Gamma_k) \tag{13}$$

Γ_k^s is for only one task, the average Γ_k^s for all K tasks can be considered as the relative efficacy factor, expressed in (14). It is worth mentioning here that the higher the value of Γ_{avg}^s, the more effective the offloading decision is.

$$\Gamma_{avg}^s = \frac{1}{K}\sum_{k=1}^{K}[(\Gamma_b - \Gamma_k) \times (\varphi_k - \Gamma_k)]. \tag{14}$$

2) Neutrality factor: The neutrality is taken as a measure of the fairness of task processing in terms of delay. The neutrality is calculated by the variance of the difference of Γ_k^s and Γ_{avg}^s, expressed in (15). It should be noted that the lower the value of the neutrality factor, the more effective the offloading decision will be.

$$\Gamma_\sigma = \frac{1}{K}\sum_{k=1}^{K}\left(\Gamma_k^s - \Gamma_{avg}^s\right)^2. \tag{15}$$

The goal of optimization is to provide maximum fairness between tasks, i.e., ensuring the minimum average processing delay of all K tasks/sub-tasks scheduled at t-time slot. Due to the influx of dynamic tasks and the complex VECN environment, the optimization objective mentioned in (12) is difficult to achieve using traditional optimization schemes. Therefore, in this article, we propose a DRL-driven DCTO scheme to determine the optimal offloading policy.

In the following sub-section, we show the integration of the complex VECN environment with the DRL method for optimal computation offloading decision making. We use the PPO model, because it is relatively new and has proven its performance in conjunction with the edge computing platforms unlike other RL algorithms (Lv et al., 2022).

4.1. DRL and the PPO

DRL is an evolution and extension of RL with deep neural network (DNN) (Li, 2017). A typical RL problem consists of an agent interacting with temporal states of the environment for a defined goal. The DRL agent grabs observations at s_t, takes action a_t on it, then the environment moves to the next state i.e., s_{t+1}, and returns the reward r_t. Both s_t and a_t belong to state space S and action space A, respectively. The agent follows $\pi(a_t|s_t)$ policy to select action a_t.

The reward is given according to the reward function $R(s_t, a_t, s_{t+1})$ and environment state action probability $P(s_{t+1}|s_t, a_t)$. This trend continues until the agent observes a termination condition. The expected accumulated reward from state s_t at t-time slot can be expressed as:

$$R_t = \sum_{\mathrm{m}=0}^{\mathrm{m}_{end}} \gamma^{\mathrm{m}} r(s_t, a_t) \tag{16}$$

here, γ is the discount factor and is a real number, from 0 to 1. The state value function $v_\pi(s)$ is the expected accumulated reward at s following policy π, given in (17). The action value function $q_\pi(s,a)$ is the expected accumulated reward after a chooses π on s, given in (18). Both functions exhibit how decent a state and state-action pair is, their relationship is given in (19).

$$v_\pi(s) = \mathbb{E}_\pi[R_t|s_t = s] \tag{17}$$

$$q_\pi(s,a) = \mathbb{E}_\pi[R_t|s_t = s, a_t = a] \tag{18}$$

$$v_\pi(s) = \sum_{a\in A} \pi(a|s)\, q_\pi(s,a) \tag{19}$$

The ultimate goal of RL is to learn the optimal policy $v_{\pi^*}(s) = \max_a v_\pi(s) = \max_a q_{\pi^*}(s,a)$, that maximizes the expected total reward at any temporal state of the state space. Both policy and value optimization is done using DNNs in DRL. The value-based DRL algorithms attempt to optimize the difference between a value function/network and a real-value function, expressed in (20). DQN and double DQN are examples of value-based DRL methods (Van Hasselt et al., 2016).

$$L^V(\theta) = \mathbb{E}_t[v_{\pi^*}(s_t) - v(s_t;\theta))]^2 \tag{20}$$

Here, $v(\cdot;\theta)$ and $v_{\pi^*}(\cdot)$ represent the value network and real-value function, respectively, and θ represents network parameters. $\mathbb{E}_t[\cdot]$ denotes the pragmatic median, over samples from a finite memory batch. On the other hand, there are other methods of DRL that approximate the policy network (parameterized policy) using policy gradient (PG) methods, such as actor–critic and REINFORCE (Degris et al., 2012). A generally used PG estimator structure is described in (21), with estimator function $\widehat{A}_t$ and π stochastic policy.

$$\nabla L^{PG}(\theta) = \mathbb{E}_t\left[\nabla\theta\, log\, \pi\,(a_t|s_t\,;\,\theta)\,\widehat{A}_t\right] \tag{21}$$

Besides, the policy-based DRL algorithms also suffer from high variance in obtaining R_t, due to the use of Monte Carlo sampling. In addition, they also face local maxima convergence of on-policy updates. The generalized advantage estimator (GAE) overcomes some of these shortcomings, and provides a compromise amid bias and variance (Schulman et al., 2016). The GAE can be expressed as (22), while ϕ is used for the bias-variance trade-off adjustment.

$$\widehat{A}_t^{GAE(\gamma,\phi)} = \sum_{t=0}^{t_{end}} (\gamma\,\phi)^{\mathrm{m}}\, \eta_{t+\mathrm{m}}^{v} \tag{22}$$

$$\eta_t^v = r_t + \gamma\, v\,(s_{t+1}\,;\,\omega) - v(s_t\,;\,\omega). \tag{23}$$

Augmenting the ability for exploration to solve local maxima, off-policy methods are introduced. PPO, introduced by OpenAI, works under off-policy learning mechanism, The objective function of PPO is defined in (24), with $r_t(\theta)$ policy probability ratio in (25) (Schulman et al., 2017).

$$L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\widehat{A}_t,\ \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\right)\right] \tag{24}$$

$$r_t(\theta) = \pi\,(a_t|s_t\,;\theta)\,/\,\pi\,(a_t|s_t\,;\theta_{old}) \tag{25}$$

The clip function in (24), bounds r_t by the interval $[1-\epsilon, 1+\epsilon]$, with the ϵ clip range control hyperparameter. Therefore, there are two probabilities for $r_t(\theta)$, the clipped and and the unclipped. The minimum of both objective probabilities is taken, to restrict the final objective to the lower bound of the unclipped objective. Following these developments in PPO, we develop the DCTO scheme adopting PPO.

4.2. MDP model formulation

Generally, an MDP is represented by a 5-tuple $\mathcal{M} = \{S, A, P, R, \gamma\}$. The role of these MDP elements in the context of our task offloading problem is described in the following subsections.

1) State space: The learning agent takes offloading decisions on the basis of these environmental states. In our case, for each ϑ_k task, we define the state space as in (26).

$$S \triangleq \{s|s = (c_k, s_k, t_k, st_k, st_o, st_e, d_k)\} \tag{26}$$

whereas, $c_k, s_k,$ andt_k are the task parameters. st_k and st_e contain information about the remaining CPU cycles required to complete the current task by the local processing unit (LPU) and the edge, respectively. st_o is the remaining amount of task data to be offloaded to the edge. d_k belongs to the wireless communication environment, i.e., the distance between the vehicle v_k and the serving RSU.

a) The local processing state st_k: For st_k state, suppose task ϑ_k is scheduled at t_1 time slot, and the vehicle v_k itself is capable of computing it. At this point, $st_k[t_1]$ is initialized to $c(\alpha s_k)$, and $\mathfrak{C}_f^l$ CPU time is assigned to compute ϑ_k, every $t \geqslant t_1$ thereafter, where $\mathfrak{C}_f^l \subseteq \mathfrak{C}_t^l, c_n$.

The value of st_k will decrease by $\mathfrak{C}_f^l$ until the execution of ϑ_k is completed, and st_k becomes 0, as expressed in (27). The condition, $st_k = 0$, implies that the vehicle v_k can process the new task, otherwise it is called busy.

$$st_k[t] = \max\left\{\left(st_k[t-1] - \mathfrak{C}_f^l\right), 0\right\},\ t > t_1 \tag{27}$$

b) The offloading state st_o: Now, for state st_o, suppose that at time slot t_2, a task needs to be offloaded such that the conditions $\mathfrak{T}_l \leqslant \varphi_k$ and $\mathfrak{d}_k = 0$ are not feasible. At this stage, the s_k^o data of task ϑ_k is needed to be uploaded to the edge. Therefore, $st_o[t_2]$ is set equals to s_k^o. After realizing the wireless communications, $(\beta s_k^o/\Re_{k,i})$ bits will transmitted from the vehicle to the edge, followed by each $t \geqslant t_2$. Consequently, st_o is reduced by $(\beta s_k^o/\Re_{k,i})$, until the uploading process is complete or it becomes 0, this process of updating the offloading state is given by:

$$st_o[t] = \max\left\{\left(st_o[t-1] - \frac{\beta s_k^o}{\Re_{k,i}}[t-1]\right), 0\right\},\ t > t_2 \tag{28}$$

c) The edge computing st_e: The functional activity of edge state st_e, is identical to the st_k. Suppose at time slot t_3, there is a task that is uploaded to the edge and is ready to be processed. Then the edge state is set to $st_e[t_3] = \mathfrak{c}(\beta s_k^o)$, immediately. At every upcoming time slot the edge server provide $\mathfrak{C}_f^e$ CPU cycles, where $\mathfrak{C}_f^e \subseteq \mathfrak{C}_t^e, c_n$, therefore, reduces the st_e by $\mathfrak{C}_f^e$, i.e., given by:

$$st_e[t] = \max\left\{\left(st_e[t-1] - \mathfrak{C}_f^e\right), 0\right\},\ t > t_3 \tag{29}$$

Since, the state space is assumed to be discrete, therefore, each element of the state space has a sufficient range of values (i.e., $0 \leqslant c_k, 0 \leqslant s_k, 0 \leqslant t_k, 0 \leqslant st_k, 0 \leqslant st_o, 0 \leqslant st_e,$ and$0 \leqslant d_k \leqslant \mathfrak{r}$). Consequently, it depicts that our MDP contains a very large state space.

2) Action space: The offloading decisions that our learning agent makes while observing the VECN environment are taken as actions. These include, $\mathcal{L}\vartheta^e, \mathcal{B}\vartheta^o, \mathcal{P}\vartheta^o$, and $\mathcal{E}\vartheta^e$ actions. Therefore, our MDP action space is the union of 4 action sets, i.e.,

$$A \triangleq \mathcal{L}^e \cup \mathcal{B}^o \cup \mathcal{P}^o \cup \mathcal{E}^e \tag{30}$$

a) Local execution action $\mathcal{L}\vartheta^e$: The action $\mathcal{L}\vartheta^e$ is performed on a given task to process it task locally. The set $\mathcal{L}^e$, holds all such actions and can be defined as:

$$\mathcal{L}^e \triangleq \left\{\mathcal{L}\vartheta_1^e, \mathcal{L}\vartheta_2^e, \ldots, \mathcal{L}\vartheta_m^e\right\} \tag{31}$$

here, $\mathcal{L}\vartheta_m^e$ $(m \in \{1, 2, \ldots, m\})$ indicates the mth task is ready for execution in the LPU's ready queue. This action can only be taken when the ready queue of LPU has space for ϑ_k task, and an idle CPU is available. Then, the processing condition is set to $\mathfrak{d}_k = \delta_k = 0$.

b) Offloading actions $\mathcal{B}\vartheta^o$, and $\mathcal{P}\vartheta^o$: Unlike $\mathcal{L}\vartheta^e$, the actions $\mathcal{B}\vartheta^o$ and $\mathcal{P}\vartheta^o$ decide to proceed for edge computing option for a given task. But, these actions can only be taken when no idle CPU is available on LPU and or the condition $\mathfrak{T}_l \leqslant \varphi_k$ is not sufficiently good. The sets of such binary $\mathcal{B}^o$ and partial $\mathcal{P}^o$ offloading actions are defined as:

$$\mathcal{B}^o \triangleq \left\{\mathcal{B}\vartheta_1^o, \mathcal{B}\vartheta_2^o, \ldots, \mathcal{B}\vartheta_m^o\right\} \tag{32}$$

$$\mathcal{P}^o \triangleq \left\{\mathcal{P}\vartheta_1^o, \mathcal{P}\vartheta_2^o, \ldots, \mathcal{P}\vartheta_m^o\right\} \tag{33}$$

Action $\mathcal{B}\vartheta_m^o$ is taken on a given mth task at the beginning of the t_2 time slot only when there is no idle CPU available on the LPU. Besides, the action $\mathcal{P}\vartheta_m^o$ is opted when the condition $\mathfrak{T}_l \leqslant \varphi_k$ is not satisfiable despite having idle CPU on LPU. A given task can be offloaded in a binary manner or partially, not the both ways. Therefore, only one action from $\mathcal{B}\vartheta_m^o$, and $\mathcal{P}\vartheta_m^o$ actions can be taken on a given mth task at t-time slot, accordingly.

c) Edge execution action $\mathcal{E}\vartheta^e$: The Offloading actions $(\mathcal{B}\vartheta^o, \mathcal{P}\vartheta^o)$ are associated with edge execution actions. Since edge execution can only be done when a given task/sub-task is completely offloaded to the edge server. The set $\mathcal{E}^e$ containing all edge computing actions is defined as:

$$\mathcal{E}^e \triangleq \left\{\mathcal{E}\vartheta_1^e, \mathcal{E}\vartheta_2^e, \ldots, \mathcal{E}\vartheta_m^e\right\} \tag{34}$$

action $\mathcal{E}\vartheta_m^e$ illustrates the mth task ready to be executed on the vehicular edge. Suppose, at the start of the t_3, task ϑ_k is completely offloaded i.e., $st_o = 0$, then agent will be able to take action $\mathcal{E}\vartheta_m^e$ on the given task.

3) Rewards: The reward function is responsible for indicating how good the state transition is. Suppose, at the t-time slot, the agent performs action a_t and moves from s_t to s_{t+1} receiving the reward r_t.

The time consumption based reward function $R(s_t, a_t, s_{t+1})$ for DCTO scheme is defined as:

$$R(s_t, a_t, s_{t+1}) \triangleq w \,.\, \Gamma_k^s(s_t, a_t, s_{t+1}) \tag{35}$$

The constant w in (35) is used to scale the reward value range. The agent interacts with the environment from state s_t while following a stochastic policy $\pi(a_t|s_t)$. Next, a Markov chain of states is obtained, i.e., $s_t, a_t, s_{t+1}, a_{t+1}, \ldots$, then the accumulated reward can be expressed as:

$$R_t = \sum_{\mathfrak{m}=0}^{\mathfrak{m}_{end}} \gamma^{\mathfrak{m}} R(s_{t+\mathfrak{m}}, a_{t+\mathfrak{m}}, s_{t+\mathfrak{m}+1}), \quad s_t \in S. \tag{36}$$

R_t is the weighted sum of the saved latency of all the tasks if $\gamma = 1$. Therefore, finding the optimum offloading policy π^* is consistent with the main objective described in the first half of the Section 4.

4.3. The DCTO scheme

Our DCTO scheme is established on the PPO algorithm. It consists of two actors (Act_1, Act_2) and one critic $Crit_1$ network, as shown in Fig. 2. The Act_1 network is used to represent the current policy π_θ, which drives the agent to relate with the VECN environment. The $Crit_1$ network evaluates the π_θ with respect to the reward assignment. The latter updates itself, via backpropagation of the loss function. The Act_2 is used to represent the old policy $\pi_{\theta_{old}}$. After training the agent a few steps, the parameter (θ) of Act_1 is used to update Act_2. By reiterating the policy updating process until the PPO is converged, we obtain a trained VECN task-offloading model established on an actor-critic framework. The following sub-sections describe the working of our DCTO scheme and the training process of the offloading agent interacting with the environment.

1) The DCTO training Algorithm: Since the DCTO scheme is built on PPO, the DNN architecture includes parameter sharing and a GAE estimator. Then the general objective function can be expressed as in 37 while substituting (22), (25) into (24).

$$L^{PPO}(\theta) = \mathbb{E}_{\mathfrak{m}}\left[L_{\mathfrak{m}}^{CLIP}(\theta) - \zeta L_{\mathfrak{m}}^{V}(\theta)\right] \tag{37}$$

ζ is the loss coefficient here. We take pragmatic mean from mini batches-of-samples after their summation for $L_{\mathfrak{m}}^{CLIP}, L_{\mathfrak{m}}^{V}$ values, instead of expectations in (20) and (24).

The PPO based training algorithm is presented in Algo.1. The DNN initialization of both actor networks is done with the same parameters. Since Act_2 belongs to policy $\pi_{\theta_{old}}$, and policy $\pi_{\theta_{old}}$ is used to sample (lines 4–10), i.e., to explore, the VECN environment. The state transition trajectories [i.e., $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_{end})$] made by following $\pi_{\theta_{old}}$ are sampled and stored into T_{ti}. The GAE advantage A_{ti} of each trajectory T_{ti} is also calculated in advance, for training efficiency. At this stage the paired (T_{ti}, A_{ti}) sampled data is stored in the cache, to be used in the optimization stage. The exploration stage is used to refine the agent's knowledge of actions for long-term reward. However, the exploitation phase is used for optimization of rewards while following greedy approach along with the estimated GAE. For optimization of π_θ, i.e., exploitation (lines 11–13), the policy parameters are updated. Considering the cached paired (T_{ti}, A_{ti}) sampled data, gradient descent Adam is used to optimize the π_θ policy in each epoch. Later, the policy $\pi_{\theta_{old}}$ is updated with the policy π_θ, and the cached data is wiped out, then the next iteration gets under way.

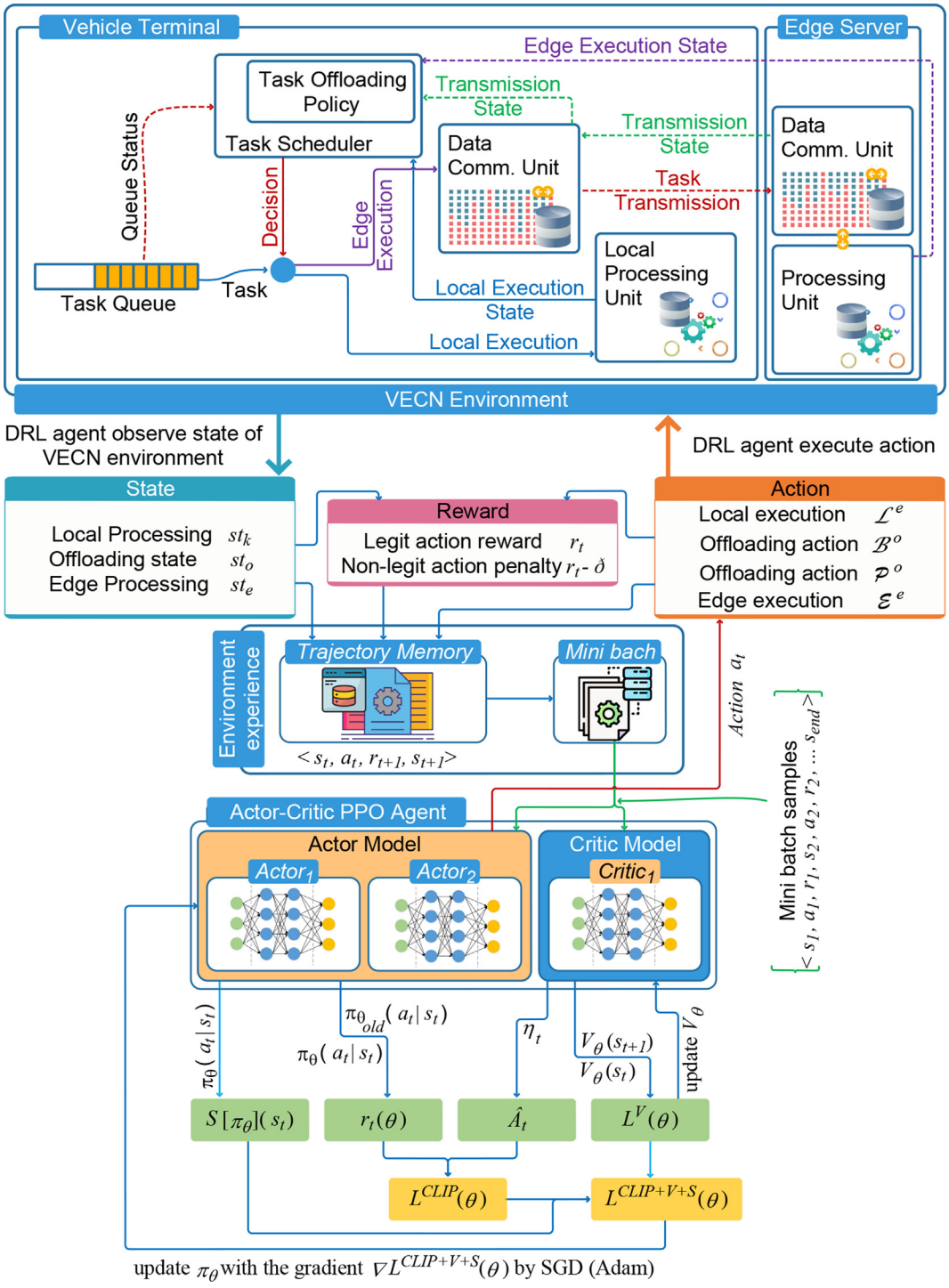


Fig. 2. DCTO agent interaction with the VECN environment.

Algorithm 1. The DCTO training algorithm

Input: $\Gamma_k^s, \Gamma_{avg}^s, \Gamma_\sigma$, st_k, st_o, st_e
Output: π_θ; where π_θ includes $\mathfrak{d}_k, \delta_k$

Initialize $\Gamma_k^s = \Gamma_{avg}^s = \Gamma_\sigma = 0$;
Initialize θ randomly for π_θ ;
Initialize the $\pi_{\theta_{old}}$ with $\theta_{old} = \theta$;
foreach *episode in episodes* **do**
  /* sampling with $\pi_{\theta_{old}}$, (i.e., exploring) */
  while $(episode \neq over)$ **do**
    Interact with VECN environment using $\pi_{\theta_{old}}$ in each t time slot, and take samples ;
    goto *Algorithm 2* and *fetch* $\Gamma_k^s, \Gamma_{avg}^s, \Gamma_\sigma$, st_k, st_o, st_e ;
    Store state transition trajectory T_{ti} ;
    Calculate advantage A_{ti} using 22 for each t time slot ;
  end
  Cache T_{ti} into set $\mathbf{T}$;
  Cache A_{ti} into set $\mathbf{A}$;
  /* Optimization of π_θ, (i.e., exploiting) */
  foreach *epoch in epochs* **do**
    Update the policy parameters according to (37), and Adam optimizer while considering $\mathbf{T}$, $\mathbf{A}$ sets.
  end
  Update parameters of $\pi_{\theta_{old}}$ with $\theta_{old} = \theta$;
  Clear $\mathbf{T}$ and $\mathbf{A}$ from the cache ;
end

Algorithm 2. DCTO algorithm

Input: Task ϑ_k's tuple, r_i, $\mathfrak{R}_{k,i}^c, \mathfrak{R}_{k,i}^{mm}, \mathfrak{S}_t^{\mathfrak{C}}, \mathfrak{t}_q, \kappa\tau$
Output: $\Gamma_k, \Gamma_k^s, \Gamma_{avg}^s, \Gamma_\sigma$, st_k, st_o, st_e

initialization: $\vartheta_k = !done$, $\Gamma_k = \Gamma_k^s = \Gamma_{avg}^s = \Gamma_\sigma = 0$
while $(\vartheta_k \neq done)$ **do**
  goto *Algorithm 1* and *fetch* $\mathfrak{d}_k$, δ_k ;
  calculate, and return Γ_k, Γ_k^s, Γ_{avg}^s, Γ_σ ;
  if $(\mathfrak{d}_k == 1 \ \& \ \delta_k == 0)$ **then**
    update, and return st_k ;
    if $(st_k == 0)$ **then**
      set $\vartheta_k = done$ and move to ϑ_{k+1} ;
  else if $\left(\mathfrak{d}_k == 0 \ \& \ \delta_k == 1 \ \& \ t_{k,i} > \frac{\beta s_k^o}{\mathfrak{R}_{k,i}}\right)$ **then**
    update, and return st_o, st_e ;
    if $(st_o == 0 \ \& \ st_e == 0)$ **then**
      set $\vartheta_k = done$ and move to ϑ_{k+1} ;
  else if $\left(\mathfrak{d}_k == 1 \ \& \ \delta_k == 1 \ \& \ t_{k,i} > \frac{\beta s_k^o}{\mathfrak{R}_{k,i}}\right)$ **then**
    update, and return st_k, st_o, st_e ;
    if $(st_k == 0 \ \& \ st_o == 0 \ \& \ st_e == 0)$ **then**
      set $\vartheta_k = done$ and move to ϑ_{k+1} ;
  end
  if $\left(t_{k,i} > \frac{s_k^{op}}{\mathfrak{R}_{k,i}} \ \& \ \delta_k == 1 \ \& \ \vartheta_k == done\right)$ **then**
    transfer output directly to v_k ;
    else
      transfer output to r_{i+1} in headway of v_k ;
    end
  end
end

a) Training optimization: Since random actions are performed in the exploration phase, the validity of the action cannot be assured, as if the agent has taken a valid action or not. According to the system model and our MDP model, there are constraints attached to each action. Suppose that a randomly chosen action is to locally process a given task ϑ_k, and there is no locally idle CPU available, then the action taken is not a legal action. Similarly, an edge computing action cannot be taken until ϑ_k is not completely uploaded to the edge server. Therefore, to improve the training, the training of the DRL agent in the sampling phase is designed in such a way that such illegal actions do not affect the environmental state. Besides, the agent is penalized over this non-legit action.

In order to force the learning agent to avoid the non-legit actions, we have added $\eth$ as a penalty term with the reward, and $\eth \in \{\eth_l, \eth_o, \eth_e\}$. $\eth_l, \eth_o$, and $\eth_e$ are the non-legit action penalties for the local, offloading, and edge processing actions, respectively. Therefore, the optimal reward function of our DCTO scheme can be rewritten as:

$$R(s_t, a_t, s_{t+1}) \triangleq w\left(\Gamma_k^s(s_t, a_t, s_{t+1}) - \eth\right) \tag{38}$$

The objective of the DCTO scheme is to train the DRL agent for offloading decision making while inquiring the attached constraints. The DRL agent interacts with the VECN environment, as shown in Fig. 2, and works according to the instructions mentioned in Sections 3 and 4.

2) The DCTO Offloading Algorithm: Each DRL agent requires interaction with the environment to explore the states. Algo.2 describes the VECN task offloading environment of our proposed DCTO scheme. Learning, and environment interaction of the DRL-agent is a joint process. Therefore, both Algo.1 and Algo.2, work for each other. Algo. 2 takes local processing or VEC processing decisions from Algo. 1. For all types of decisions, Algo.2 must calculate $\Gamma_k, \Gamma_k^s, \Gamma_{avg}^s$, and Γ_σ, and return these values to the Algo. 1.

However, if the local processing is chosen (step 5), Algo.2 updates the local state st_k and returns the updated state to the Algo.1. Besides, if a binary offloading decision is made (step 9), the Algo.2 updates the states st_o, st_e, accordingly. Also, if a partial offloading decision is given (step 13), then the states st_k, st_o, and st_e are updated, and returned to the Algo.1. The contact period of the vehicle v_k with r_i is considered before the latency calculation and updating process for binary and partial offloading decisions. The Algo.1 assigns rewards or penalties to the learning agent based on the states st_k, st_o, st_e at time slot $t+1$ following the rules described in Table 1.

Later, if task states st_k, st_o, and st_e all are set equal to 0 and then the task ϑ_k's status is set to $\vartheta_k == done$, it is considered that the

Table 1
Reward and penalty mechanism.

	$\mathfrak{d}_k$	δ_k	$\mathfrak{T}_l \leqslant \varphi_k$	$\mathfrak{T}_e \leqslant \varphi_k$	$t_{k,i} > (\beta s_k^o/\mathfrak{R}_{k,i})$	**Remark**
	0	0	-	-	-	No task
local processing	1	0	T	-	-	Reward
	1	0	F	-	-	Penalty
Binary offloading	0	1	-	T	T	Reward
	0	1	-	F	T	Penalty
	0	1	-	T	F	Penalty
	0	1	-	F	F	Penalty
Partial offloading	1	1	T	T	T	Reward
	1	1	F	T	T	Penalty
	1	1	T	F	T	Penalty
	1	1	F	F	T	Penalty
	1	1	T	T	F	Penalty
	1	1	F	T	F	Penalty
	1	1	T	F	F	Penalty
	1	1	F	F	F	Penalty

current task has been processed successfully. Then it is handed over to the result dissemination process, and the next ϑ_{k+1} task is loaded for processing. The results dissemination process then observes the contact duration $t_{k,i}$ of vehicle v_k with r_i, the resulting output of task s_k^{op}, current data rate $\Re_{k,i}$, edge offloading decision parameter, and the task processing status ϑ_k. If the vehicle v_k's stay time under RSU r_i is greater than the result transfer time (i.e., $s_k^{op}/\Re_{k,i}$), and the offloading option is opted (i.e., $\delta_k == 1$), and the task processing is completed (i.e., $\vartheta_k == done$), as per step 18, then the result is sent directly to v_k. Otherwise, the result is forwarded to the nearest r_{i+1} in headway of v_k.

After exploring the VECN environment and exploiting the sample state, action, and reward trajectories, the learning converges to a certain point. At this particular point the DCTO scheme is said to be optimally trained. The following section provides a detailed discussion of the results obtained for testing our DRL agent.

5. Simulation setup, results and discussion

In this section, extensive simulation evaluations are presented to estimate the proposed DCTO scheme.

5.1. Simulation setup

We consider a scenario with a one-way straight 2 km long road, with evenly distributed 5 RSUs, each having one sided coverage range of 200 m for C-V2X, and 150 m for mmWave. Similarly, the vehicles have 100 m C-V2X, and 80 m mmWave communication range. We assume that each vehicle is moving on the road with a random speed between 8 m/s – 18 m/s. The task size and processing threshold time is randomly selected from a range of 1 – 50 MB, and 200 – 500 ms, respectively. Moreover, we have also implemented a sub-task mechanism. The sub-task ratio is set to $1 : [1 - 100]$, i.e., a single task may contain 1 to 100 sub-tasks. Simulations are implemented in Python 3.7 using PyTorch STable (1.13.1) for all the schemes. The DRL algorithms used in these schemes are implemented on the guidelines provided by OpenAI. Other simulation parameters and DRL hyperparameters are listed in the Table 2.

5.2. Contender approaches and evaluation metrics

The DCTO scheme has two variants. Both variants are extensively evaluated under highly dynamic contexts against other DRL based offloading schemes as follows. To ensure a fair comparison, the simulation parameters were kept consistent across all schemes, with any exceptions noted.

- *DCTO-PPO*: It is the exact proposed scheme.
- *DCTO-TRPO*: It differs from DCTO-PPO with the change of DRL algorithm, that the learning agent is based on trust region policy optimization (TRPO).
- *DQN*: It is a Deep Q-Network (DQN) based MEC task offloading scheme where vehicles process a task locally or offload to the MEC. This scheme optimizes the computation latency while considering binary offloading and using LTE-C-V2X RAT (Zhang et al., 2022c).
- *DFO-DDQN*: It is another LTE-C-V2X based MEC offloading scheme based on dueling DQN. DFO-DDQN considers partial task offloading and optimizes the latency while including local and V2I computation offloading scenarios (Tang et al., 2022).
- *AUC-AC*: It is an edge based task offloading scheme and uses actor critic (A2C) DRL algorithm for decision making. It is a C-V2X RAT based binary offloading scheme that maximizes the saved latency (Gu et al., 2021).

The episodic reward and the algorithmic time frames per step (TFPS) are used to evaluate the DRL properties of learning agents. Additionally, we use the average efficacy and average neutrality factors to evaluate communication and computation delays, and the task success and drop ratio to measure the number of completed tasks in the VECN environment.

Table 2
Simulation parameters.

Parameter	*Value*
No. of vehicles v_k and RSUs	20, 4
Task generation probability	0.7
One bit processing density	300 CPU cycles
Per CPU Computation capacity of v_k and r_i	1 MHz–1 GHz, 1.5 GHz–2.5 GHz
p_k, h, and σ	1.3w, 0.23, $3 * 10^{-13}$
$p_k^{mm}, \sigma_{mm}, \varsigma$, and $\mathcal{G}_{r_x,t_x}$	30 dB, −174, 3.3, 13 dB
Cellular Bandwidth	20 MHz in 800 MHz frequency range
mmWave Bandwidth	200 MHz in 28 GHz frequency range
DRL Parameter	
Training/ Evaluation time steps	100 k
per update steps	2048
Learning rate, γ / discount factor	0.0007, 0.99
Batch size, epochs	64, 10
GAE ϕ, Clip range, w	0.9–1, 0.2, 0.2
Entropy, value function coefficient	0–0.01, 0.5–1
Max value of gradient clipping	0.5
Penalty $\eth_l, \eth_o$, and $\eth_e$	4

5.3. Results and discussion

We have conducted several evaluations to test the validity and robustness of our proposed DCTO scheme under a highly variable and dynamic VECN environment.

1) DRL performance evaluation: In this section, we evaluate the DRL features of our proposed DCTO schemes.

Figs. 3(a) and 3(b) depict the effect of varying the learning rate on the episodic reward in DRL. Fig. 3(c) shows the variation of TFPS during the learning process with different learning rates. The learning rates is set to 0.0005, 0.0007, and 0.0009 and the mean episodic reward rate is observed. Fig. 3(a) shows the actual rewards but it is difficult to determine which learning rate produces the best rewards. To obtain a clearer picture, the Savitzky-Golay filter was applied to smooth the trend of the obtained rewards, as shown in Fig. 3(b). From the Fig. 3(b), it can be seen that a learning rate of 0.0007 yields a slightly higher reward rate than the other learning rates.

In the DRL based task offloading scheme, the TFPS rate is a crucial metric to evaluate the performance of the DRL agent. The aim is to achieve a lower TFPS over time, indicating that the agent is performing well. The results of our experiments, depicted in Fig. 3(c), show that the learning rate of 0.0007 resulted in a better TFPS compared to the learning rates of 0.0005 and 0.0009. Therefore, the learning rate of 0.0007 is used in all further experiments for optimal performance in DCTO schemes for.

Extending the DRL characteristics evaluations, we have compared DCTO scheme with other DRL based VECN task offloading solutions, as shown in Figs. 4(a–c). Episodic reward is typically related to communication and processing delays in most task-offloading environments (Zhang et al., 2022c; Tang et al., 2022; Gu et al., 2021). Therefore, we have taken the saved latency as the episodic reward for equitable comparisons as mentioned in Section 4.3. Figs. 4(a) and 4(b) display the mean episodic rewards of various schemes. As evident, DCTO_PPO and DCTO_TRPO outperform other DRL algorithms in the VECN environment in terms of rewards. The average reward achieved by DCTO_PPO is 41.40, while DCTO_TRPO achieved 38.81. Both DCTO_PPO and DCTO_TRPO are close to each other as they are designed over Minorize-Maximization (MM), but DCTO_PPO

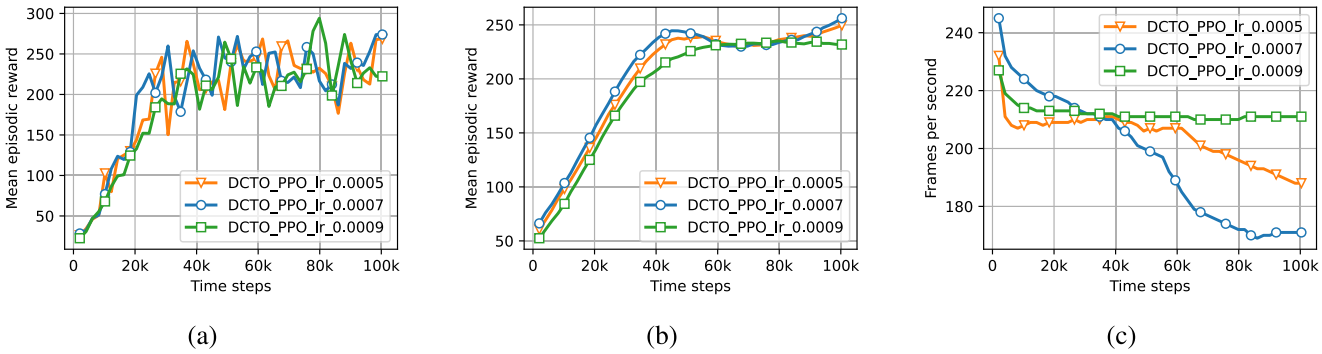


Fig. 3. Impact of variable learning rate on mean episodic reward, and TFPS variability.

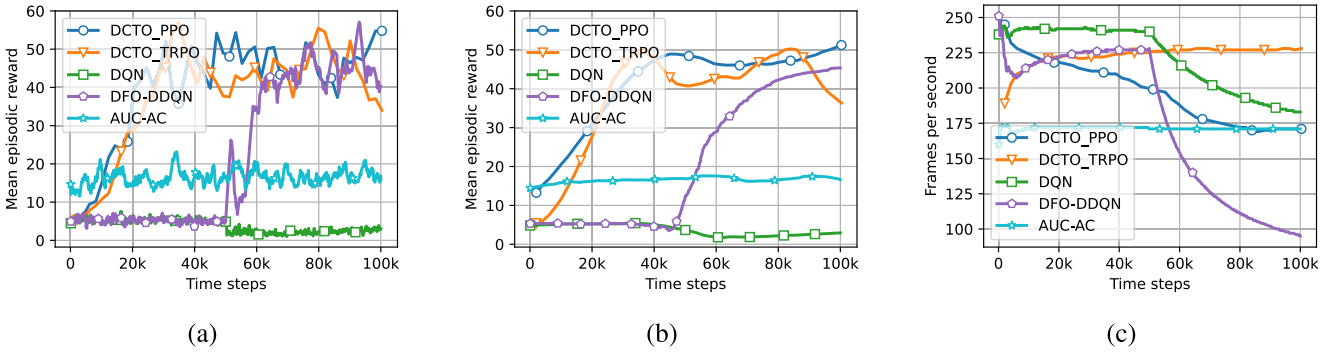


Fig. 4. Mean episodic reward and TFPS analysis of different DRL algorithms.

takes the lead. This lead of DCTO_PPO is due to it's simplicity, less complexity, and efficient sampling mechanism. However, TRPO uses KL divergence and trust region constraints, which increase its complexity and affect convergence. In fact, PPO is a simpler and less complex variant of TRPO, which uses a clipping and penalty mechanism instead of KL divergence. On the other hand, similar to the DCTO schemes, the AUC-AC also employs the actor-critic model. In terms of rewards, the AUC-AC performed better than DQN and DFO-DDQN and obtained an average reward of 16.92, but it did not reach the level of performance of both DCTO schemes. The DQN algorithm had an average reward of 03.43, while DFO-DDQN was able to attain an average reward of 11.10. Besides, if we compare these actor critic based solutions with the Q-Network based solutions, they performed worse in comparison. Both DQN and DFO-DDQN overlap each others half the time but then DFO-DDQN takes the lead over DQN. The DQN scheme remain unable to recover itself until the end of the simulation. Besides, DFO-DDQN manages to recover itself in the second half of the simulation, but is still unable to compete in performance with the DCTO schemes. In addition to the reward comparison, Fig. 4(a) depicts the analysis of TFPS for all algorithms. DCTO_PPO starts off with a high TFPS but gradually learns to maintain a lower TFPS, ultimately reaching an average TFPS of 196.67. On the other hand, DCTO_TRPO begins with a low TFPS but over time, instead of decreasing, it increases. However, after a certain period it stabilizes at almost the same TFPS for each time frame and has an average TFPS of 223.12. The learning behavior of AUC-AC is similar to that of DCTO_TRPO, but with a tendency to have a flat slope close to zero. It starts with a lower TFPS, increases, and then stabilizes at almost the same TFPS for each time step, although lower than DCTO_TRPO. AUC-AC reaches an average TFPS of 171.29. In comparison, DQN and DFO-DDQN exhibit a different but similar learning pattern. Initially, the slope of DQN is close to zero, but as learning starts, it maintains a lower TFPS, ending up inferior to all but DCTO_TRPO. DFO-DDQN is superior in terms of TFPS, but falls short of AUC-AC and the proposed DCTO_PPO over half of the time. DQN and DFO-DDQN reach average TFPS values of 217.54 and 206.69, respectively.

Summary: DRL algorithms have distinct features and performance measures, but their effectiveness can vary in different settings. In our study, we examined the characteristics of DRL in the 5G-NR-V2X VECN environment. Among the analyzed algorithms, DCTO schemes showed the most consistent reward, with DCTO_PPO performing particularly well. While DFO-DDQN had the lowest TFPS rate, it was not as consistent as DCTO_PPO, which consistently achieved lower TFPS from beginning to the end of the evaluation. To conclude, the PPO-based DCTO algorithm demonstrated the best overall performance among the compared schemes.

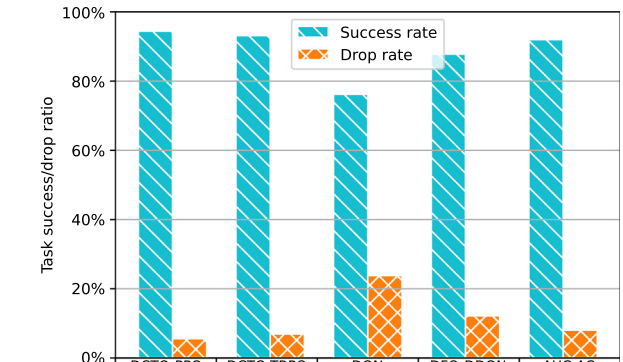
2) Offloading performance evaluation: In this section, we assess the offloading performance of our proposed DCTO schemes compared to other state-of-the-art methods. Task turnover vs drop ratio and latency are important metrics to consider when evaluating the effectiveness of an offloading scheme.

Fig. 5, provides an analysis regarding task success rate versus task drop rate of various task offloading schemes. This evaluation is part of the evaluation process for deciding whether to compute the task locally or offload it to the VEC server. Timely and accurate decision making according to the task and resource constraints leads to successful task execution or else it may lead to failure. On similar grounds, both variants of DCTO outperform all other task offloading schemes in terms of task success and failure rates. DCTO_PPO provides task processing success and failure rates of 94.43%, 05.57%, respectively. The other variant, DCTO_TRPO provides a success and drop rate of 93.12%, 06.88%, respectively.

In contrast to the DCTO schemes, DQN, DFO-DDQN, and AUC-AC provide task processing success rates of 76.22%, 87.82%, and 91.99%, respectively. Similarly, the task drop rates of DQN, DFO-DDQN, and AUC-AC are 23.78%, 12.18%, and 08.01%, respectively. Statistically, AUC-AC performs well against DQN and DFO-DDQN schemes but not against DCTO schemes. DFO-DDQN maintains its

superiority over the DQN based offloading scheme, but is inferior to all other schemes.

Similarly, minimizing task processing latency is also considered important in any task offloading scheme. We consider efficacy and neutrality factors as performance indicators, which are derived from task processing latencies and are described in Sections 4-1 and 4-2. Fig. 6 and Fig. 7 show average relative efficacy and neutrality statistics for increasing number of tasks.



	DCTO-PPO	DCTO-TRPO	DQN	DFO-DDQN	AUC-AC
Success rate	94.43 %	93.12 %	76.22 %	87.82 %	91.99 %
Drop rate	05.57 %	06.88 %	23.78 %	12.18 %	08.01 %

Fig. 5. Performance of DCTO and other offloading schemes with respect to task success and drop (failure) rate.

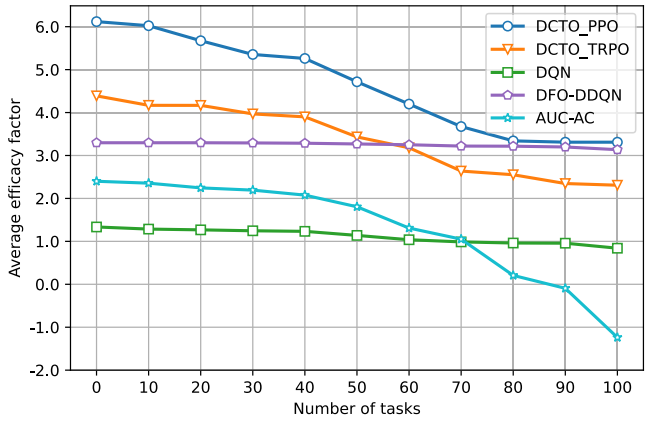


Fig. 6. Average relative efficacy of task offloading schemes against increasing number of tasks.

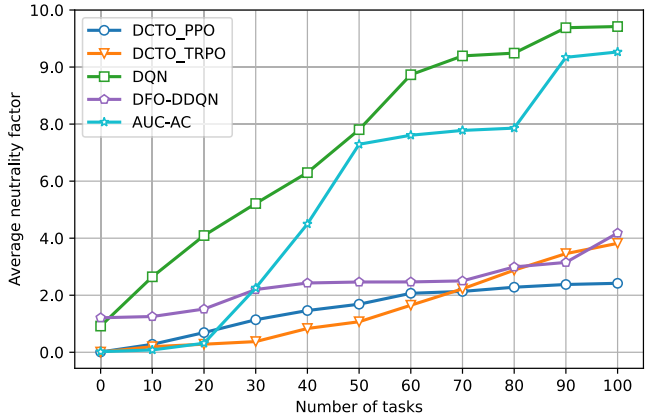


Fig. 7. Average relative neutrality of task offloading schemes against increasing number of tasks.

We evaluate the efficacy and neutrality factor of over 100 tasks per second with 1 to 100 random subtasks for each task. For fair comparison, all the parameters of this test are kept as mentioned in Section 5.1. From the Figs. 6 and 7, it can be seen that each task offloading scheme is able to save some time but with performance differences. However, both of DCTO schemes perform almost identically but with different efficacy and neutrality factors, and outperform to other schemes. All the offloading schemes initially has better save time ratio, when task rate is up to 40 tasks per second. Since, the system will perform better when the number of tasks is less.

Conversely, if the tasks are increased, the performance of the system will ultimately depend on the task handling mechanism and its capacity. But both DCTO schemes in this range achieve much better average efficacy and neutrality factors due to the binary offloading mechanism. When the task rate is increased from 40 to 80 tasks per second, DCTO schemes perform both binary and partial offloading. As the task rate goes up from 40 to 80 tasks per second, the DCTO schemes engage in both binary and partial offloading. And when the task rate keeps rising from 80 to 100 tasks per second, the DCTO schemes fully adopt partial offloading, achieving optimal efficacy and neutrality factors while balancing the computing platforms. On average, DCTO_PPO attained an efficacy factor of 4.63, and DCTO_TRPO attained 3.36, with average neutrality factor of 1.29, and 1.31, respectively. Additionally, DCTO_PPO saved 44.87% of time, while DCTO_TRPO managed to saved 43.59% of time. The DFO-DDQN algorithm balances the computing platforms through partial offloading, but it does not achieve the same level of time savings as the DCTO schemes. On average, it achieved an efficacy factor of 3.25 and a neutrality factor of 2.05, while saving 36.53% of the time. DQN and AUC-AC both perform binary offloading, but DQN performs better in terms of average efficacy compared to AUC-AC. On the other hand, AUC-AC performs better in terms of average neutrality compared to DQN. Both DCTO schemes and DFO-DDQN are similar in terms of neutrality factor, with DCTO_PPO outperforming all other offloading schemes. The average efficacy factor achieved by DQN and AUC-AC is 1.11 and 1.30, respectively, and their average neutrality factor is 6.28 and 4.58, respectively. On average, DQN and AUC-AC saved 20.72% and 41.10% time, respectively.

Task threshold time is another important constraint when evaluating task offloading schemes. In this study, we evaluated the performance of all tasks offloading schemes under various task threshold times, as depicted in Figs. 8 and 9. The results varied due to randomness, but all offloading schemes were able to process tasks even at the lowest φ_k. However, their performance in terms of efficacy and neutrality factors varies. DCTO schemes initially have the lowest average efficacy factor because the randomly assigned VEC CPU frequency is lower than that of the other schemes. However, as φ_k increases, both DCTO schemes outperform the other schemes in terms of efficacy and neutrality. The DCTO_PPO and DCTO_TRPO achieved an average efficacy factor of 4.70 and 4.14, and an average neutrality factor of 2.40 and 2.66, respectively. Additionally, these managed to save 37.55% and 38.96% of time, respectively. AUC-AC performs better than DQN and DFO-DDQN in terms of average efficacy, but not as well as the DCTO schemes. DFO-DDQN also performs well compared to DQN in both efficacy and neutrality factors. The average efficacy factor for the DQN, DFO-DDQN, and AUC-AC schemes is 2.29, 3.50, and 4.17, respectively. The average neutrality factor for these schemes is 3.26, 0.97, and 3.21, respectively. Additionally, these schemes saved an average of 27.25%, 27.61%, and 34.38% of time, respectively. Initially DCTO_PPO, DQN, DFO-DDQN follow the same learning pattern for efficacy factor, up to $\varphi_k = 30$ ms. Later, the DCTO_PPO took over until it achieve the maximum efficacy factor as per the system configuration. The same applies to the neutrality factor, as it is derived from the efficacy factor. Moreover, DCTO schemes follow a partial offloading mechanism when φ_k is set to

a minimum. As φ_k gradually increases to 200 ms, the DCTO schemes follow a combination of partial and binary offloading mechanisms. Later, beyond 200 ms DCTO schemes switch to binary offloading mechanism to have maximum efficacy and neutrality factor. It is also observed that, if φ_k is too small, the task drop ratio increases for all schemes. Nevertheless, DCTO_PPO manages to control the task drop ratio through its partial offloading mechanism.

Figs. 10, 11, show the analysis of average efficacy and neutrality behavior against different tasks sizes. The task processing pattern

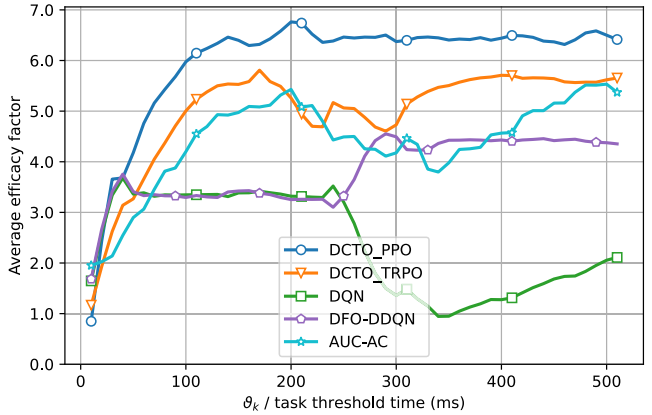


Fig. 8. Average relative efficacy of task offloading schemes against increasing task processing threshold time.

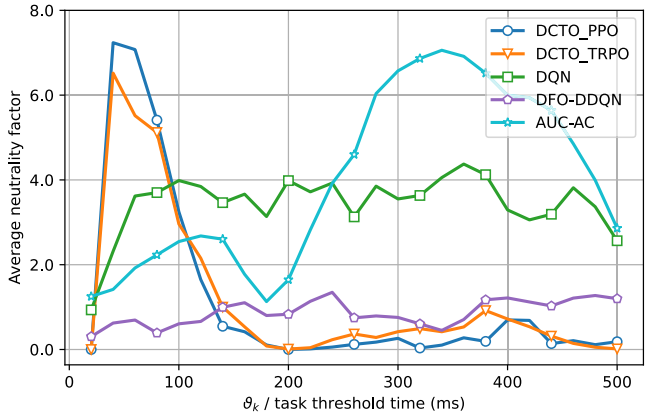


Fig. 9. Average relative neutrality of task offloading schemes against increasing task processing threshold time.

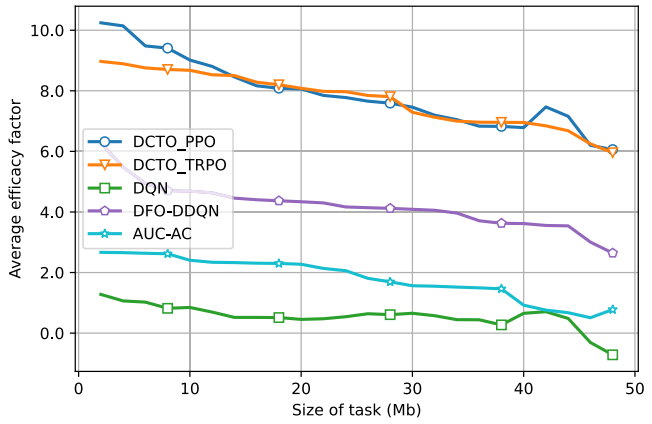


Fig. 10. Average relative efficacy of task offloading schemes against increasing task size.

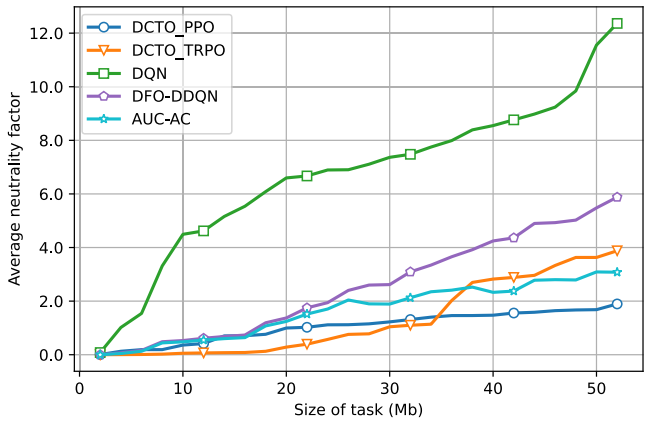


Fig. 11. Average relative neutrality of task offloading schemes against increasing task size.

of all the offloading schemes is almost the same, i.e., as the task size increases, the saving time decreases. This behavior is observable from both figures. On average, DCTO_PPO and DCTO_TRPO achieve an average efficacy of 7.75 and 7.48, respectively, and an average neutrality of 1.14 and 1.50, respectively. Additionally, these managed to save 41.04% and 39.04% of time, respectively. The AUC-AC scheme resulted in an average efficacy and neutrality of approximately 1.76 and 1.50, respectively, and saved 40.34% of time. Meanwhile, DQN and DFO-DDQN ended up with average efficacy of 0.54 and 4.18, respectively, and average neutrality of 7.77 and 2.74, respectively, and saved 26.17% and 27.69% of time, respectively. Analyzing these results, the DQN offloading scheme handles it the worst, as it manages to save much less time from start to finish than all other schemes. AUC-AC manages it well compared to DQN, and DFO-DDQN outperforms both the DQN and AUC-AC schemes. On the other hand, both DCTO schemes outperform all offloading schemes in terms of average efficacy and neutrality factors. As the major performance factor of DCTO schemes, we consider 5G-NR-V2X based cellular and mmWave RATs for V2I communication. The combined use of these RATs increases the performance of the V2I link, therefore, providing low-latency transmission. In contrast, all other schemes only consider C-V2X RATs for V2I links, which leads to delayed transmission compared to 5G-NR-V2X RATs.

Summary: In a given VECN environment, all the task offloading schemes were able to perform effectively with varying degrees of success. AUC-AC and DQN only operate with a binary offloading mechanism, which results in degradation in delay control when the task size increases or the task threshold time decreases. In contrast, DFO-DDQN operates with both binary and partial offloading mechanisms, which contributes to its superiority over AUC-AC and DQN in most cases. Another factor affecting the performance of the listed schemes is that they are all based on LTE-C-V2X RATs. However, the use of 5G-NR-V2X RATs and binary and partial offloading mechanisms is also a performance-enhancing factor for the DCTO schemes. Based on the results, the DCTO_PPO scheme is the clear winner across all metrics, including task success or drop ratio, average efficacy, average neutrality, and the percentage of time saved.

6. Conclusion

In this paper, we presented a study on the performance enhancement of VECNs. A DRL-based DCTO task offloading scheme is developed considering 5G-NR-V2X heterogeneous RATs, and both binary and partial offloading mechanisms. The DCTO scheme

is designed for a highly random and complex VECN environment while considering the relative efficacy and neutrality factors. The DCTO scheme opportunistically switches its RATs and offloading mechanisms while targeting delay minimization. This switching mechanism works by considering the V2I contact duration and vehicle's mobility headway. The switching mechanism is motivated by the task offloading decision mechanism, and the decision mechanism is controlled by the DRL agent. Additionally, the DRL agent is trained to focus on delay optimization. The results of extensive evaluations show that the PPO based DCTO scheme significantly improves task success rate, increasing it from 2.61% to 21.34%. Additionally, the efficacy factor is improved from 1.38 to 3.52 and the neutrality factor is reduced from 4.99 to 0.76. The average task processing time is also reduced by a range of 3.77% to 24.15%. Furthermore, the DCTO_PPO scheme also achieved significant results in terms of reward and TFPS ratio compared to the other evaluated schemes. In future work, we plan to incorporate V2V offloading, energy consumption, and load balancing using a multi-agent DRL mechanism.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

3GPP, 2019. Study on evaluation methodology of new vehicle-to-everything v2x use cases for lte and nr (release 15). 3gpp rel 15, no. TR 37.885.

Ahmed, M., Raza, S., Mirza, M.A., Aziz, A., Khan, M.A., Khan, W.U., Li, J., Han, Z., 2022. A survey on vehicular task offloading: Classification, issues, and challenges. J. King Saud Univ.-Compu. Informat. Sci. 34 (7), 4135–4162.

Boukerche, A., Sotoro, V., 2020. Computation offloading and retrieval for vehicular edge computing: Algorithms, model and classification. ACM Comput. Surv. (CSUR) 53 (4), 1–35.

Chen, C., Zeng, Y., Li, H., Liu, Y., Wan, S., 2022a. A multi-hop task offloading decision model in mec-enabled internet of vehicles. IEEE Internet Things J. 1–1.

Chen, C., Li, H., Li, H., Fu, R., Liu, Y., Wan, S., 2022b. Efficiency and fairness oriented dynamic task offloading in internet of vehicles. IEEE Trans. Green Commun. Network. 6 (3), 1481–1493.

Cui, Y., Du, L., Wang, H., Wu, D., Wang, R., 2021. Reinforcement learning for joint optimization of communication and computation in vehicular networks. IEEE Trans. Vehicular Technol. 70 (12), 13062–13072.

Degris, T., White, M., Sutton, R.S., 2012. Linear off-policy actor-critic. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012. icml.cc/ Omnipress.

Deng, X., Sun, Z., Li, D., Luo, J., Wan, S., 2021. User-centric computation offloading for edge computing. IEEE Internet Things J. 8 (16), 12559–12568.

Du, J., Sun, Y., Zhang, N., Xiong, Z., Sun, A., Ding, Z., 2022. Cost-effective task offloading in noma-enabled vehicular mobile edge computing. IEEE Syst. J.

Gu, L., Xu, X., Qi, L., Zhang, Y., Zhang, X., Dou, W., 2021. Cooperative task offloading for internet of vehicles in cloud-edge computing. In: 2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys). IEEE, 2021, pp. 1537–1544.

ITU, 2021. Land mobile (including wireless access) - volume 4: Intelligent transport systems. In: ITU-R WP5A. Radiocommunication Bureau, ITU, March 2021. [Online]. Available: http://handle.itu.int/11.1002/pub/81734039-en.

Jiang, X., Yu, F.R., Song, T., Leung, V.C., 2021. Resource allocation of video streaming over vehicular networks: a survey, some research issues and challenges. IEEE Trans. Intell. Transp. Syst. 23 (7), 5955–5975.

Jin, H., Gregory, M.A., Li, S., 2022. A review of intelligent computation offloading in multi-access edge computing. IEEE Access 10, 71481–71495.

Khan, W.U., Ihsan, A., Nguyen, T.N., Ali, Z., Javed, M.A., 2022. Noma-enabled backscatter communications for green transportation in automotive-industry 5.0. IEEE Trans. Industr. Inf. 18 (11), 7862–7874.

Li, Y., 2017. Deep reinforcement learning: An overview, arXiv preprint arXiv:1701.07274.

Liu, S., Yu, J., Deng, X., Wan, S., 2021. Fedcpf: An efficient-communication federated learning approach for vehicular edge computing in 6g communication networks. IEEE Trans. Intell. Transp. Syst. 23 (2), 1616–1629.

Liu, J., Ahmed, M., Mirza, M.A., Khan, W.U., Xu, D., Li, J., Aziz, A., Han, Z., 2022. RL/DRL meets vehicular task offloading using edge and vehicular cloudlet: A survey. IEEE Internet Things J. 9 (11), 8315–8338.

Luo, Q., Li, C., Luan, T., Shi, W., 2021. Minimizing the delay and cost of computation offloading for vehicular edge computing. IEEE Trans. Services Comput.

Lv, P., Xu, W., Nie, J., Yuan, Y., Cai, C., Chen, Z., Xu, J., 2022. Edge computing task offloading for environmental perception of autonomous vehicles in 6g networks. In: IEEE Trans. Network Sci. Eng., pp. 1–18.

Naik, G., Choudhury, B., Park, J.-M., 2019. IEEE 802.11 bd & 5G NR V2X: Evolution of radio access technologies for V2X communications. IEEE Access 7, 70169–70184.

Nguyen, K., Drew, S., Huang, C., Zhou, J., 2022. Parked vehicles task offloading in edge computing. IEEE Access 10, 41592–41606.

Qiao, F., Wu, J., Li, J., Bashir, A.K., Mumtaz, S., Tariq, U., 2020. Trustworthy edge storage orchestration in intelligent transportation systems using reinforcement learning. IEEE Trans. Intell. Transp. Syst. 22 (7), 4443–4456.

Raza, S., Wang, S., Ahmed, M., Anwar, M.R., Mirza, M.A., Khan, W.U., 2021. Task offloading and resource allocation for iov using 5g nr-v2x communication. IEEE Internet Things J. 9 (13), 10 97–10410.

Raza, S., Ahmed, M., Ahmad, H., Mirza, M.A., Habib, M.A., Wang, S., 2022. Task offloading in mmwave based 5g vehicular cloud computing. J. Ambient Intell. Humanized Comput., 1–13

Schulman, J., Moritz, P., Levine, S., Jordan, M.I., Abbeel, P., 2016. High-dimensional continuous control using generalized advantage estimation. In: Bengio, Y., LeCun, Y. (Eds.), 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2–4, 2016, Conference Track Proceedings.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O., 2017. Proximal policy optimization algorithms. CoRR, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347.

Shibata, Y., Sakuraba, A., Sato, G., Uchida, N., 2019. Iot based wide area road surface state sensing and communication system for future safety driving. In: International Conference on Advanced Information Networking and Applications, Matsue, Japan, March 2019, pp. 1123–1132.

Shuai, R., Wang, L., Guo, S., Zhang, H., 2021. Adaptive task offloading in vehicular edge computing networks based on deep reinforcement learning. 2021 IEEE/CIC International Conference on Communications in China (ICCC). IEEE, Xiamen, China.

Shu, W., Li, Y., 2022. Joint offloading strategy based on quantum particle swarm optimization for mec-enabled vehicular networks. Digital Commun. Networks.

Tang, F., Mao, B., Kato, N., Gui, G., 2021. Comprehensive survey on machine learning in vehicular network: technology, applications and challenges. IEEE Commun. Surv. Tutor. 23 (3), 2027–2057.

Tang, H., Wu, H., Qu, G., Li, R., 2022. Double deep q-network based dynamic framing offloading in vehicular edge computing. IEEE Trans. Network Sci. Eng.

Van Hasselt, H., Guez, A., Silver, D., 2016. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, no. 1, Phoenix, Arizona USA.

Wang, Y., Sheng, M., Wang, X., Wang, L., Li, J., 2016. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Trans. Commun. 64 (10), 4268–4282.

Wang, H., Li, X., Ji, H., Zhang, H., 2018. Federated offloading scheme to minimize latency in mec-enabled vehicular networks. 2018 IEEE Globecom Workshops (GC Wkshps). IEEE, pp. 1–6.

Wang, D., Song, B., Lin, P., Yu, F.R., Du, X., Guizani, M., 2022. Resource management for edge intelligence (ei)-assisted iov using quantum-inspired reinforcement learning. IEEE Internet Things J. 9 (14), 12588–12600.

Yao, L., Xu, X., Bilal, M., Wang, H., 2022. Dynamic edge computation offloading for internet of vehicles with deep reinforcement learning. IEEE Trans. Intell. Transport. Syst., 1–9

Ye, Q., Shi, W., Qu, K., He, H., Zhuang, W., Shen, X., 2021. Joint ran slicing and computation offloading for autonomous vehicular networks: A learning-assisted hierarchical approach. IEEE Open J. Vehicular Technol. 2, 272–288.

Zhang, S., Gu, H., Chi, K., Huang, L., Yu, K., Mumtaz, S., 2022a. Drl-based partial offloading for maximizing sum computation rate of wireless powered mobile edge computing network. IEEE Trans. Wireless Commun. 21 (12), 10934–10948.

Zhang, Q., Wen, H., Liu, Y., Chang, S., Han, Z., 2022b. Federated reinforcement learning enabled joint communication, sensing and computing resources allocation in connected automated vehicles networks. IEEE Internet Things J. 12. 1-1.

Zhang, L., Zhou, W., Xia, J., Gao, C., Zhu, F., Fan, C., Ou, J., 2022c. Dqn-based mobile edge computing for smart internet of vehicle. EURASIP J. Adv. Signal Process. 2022 (1), 1–16.

Zhou, Z., Wang, Z., Yu, H., Liao, H., Mumtaz, S., Oliveira, L., Frascolla, V., 2020. Learning-based urllc-aware task offloading for internet of health things. IEEE J. Sel. Areas Commun. 39 (2), 396–410.