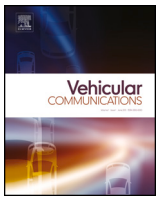
Vehicular Communications 51 (2025) 100862

Contents lists available at ScienceDirect

Vehicular Communications

journal homepage: www.elsevier.com/locate/vehcom



Offloading in V2X with road side units: Deep reinforcement learning

Widhi Yahya [a,*], Ying-Dar Lin [b], Faysal Marzuk [c], Piotr Chołda [c], Yuan-Cheng Lai [d]

[a] Department of Computer Science, Universitas Brawijaya, Malang 65145, Indonesia
[b] Department of Computer Science, National Yang Ming Chiao Tung University, Hsinchu 300, Taiwan
[c] Institute of Telecommunications, AGH University of Krakow, Kraków 30-059, Poland
[d] Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan

ARTICLE INFO

Keywords:
5G
AI
Mobile edge computing
Offloading
Reinforcement learning
V2X

ABSTRACT

Traffic offloading is crucial for reducing computing latency in distributed edge systems such as vehicle-to-everything (V2X) networks, which use roadside units (RSUs) and access network mobile edge computing (AN-MEC) with ML agents. Traffic offloading is part of the control plane problem, which requires fast decision-making in complex V2X systems. This study presents a novel ratio-based offloading strategy using the twin delayed deep deterministic policy gradient (TD3) algorithm to optimize offloading ratios in a two-tier V2X system, enabling computation at both RSUs and the edge. The offloading optimization covers both vertical and horizontal offloading, introducing a continuous search space that needs fast decision-making to accommodate fluctuating traffic in complex V2X systems. We developed a V2X environment to evaluate the performance of the offloading agent, incorporating latency models, state and action definitions, and reward structures. A comparative analysis with metaheuristic simulated annealing (SA) is conducted, and the impact of single versus multiple offloading agents with deployment options at a centralized central office (CO) is examined. Evaluation results indicate that TD3's decision time is five orders of magnitude faster than SA. For 10 and 50 sites, SA takes 602 and 20,421 seconds, respectively, while single-agent TD3 requires 4 to 24 milliseconds and multi-agent TD3 takes 1 to 3 milliseconds. The average latency for SA ranges from 0.18 to 0.32 milliseconds, single-agent TD3 from 0.26 to 0.5 milliseconds, and multi-agent TD3 from 0.22 to 0.45 milliseconds, demonstrating that TD3 approximates SA performance with initial training.

1. Introduction

Fifth generation (5G) networks facilitate the growth of intelligent transportation systems (ITS) with high capacity and low latency communication, enhancing transportation efficiency, safety, and comfort [1]. Vehicles, equipped with computing devices, communication tools, and embedded sensors, collect data for artificial intelligence (AI) agents with machine learning (ML) algorithms to perform ITS services. In services such as AI-assisted driving, collision detection, and vehicle platooning, vehicles interact with environmental entities, referred to as vehicle-to-everything (V2X), including scenarios such as vehicle-to-vehicle (V2V), vehicle-to-pedestrian (V2P), vehicle-to-roadside unit (V2R), and vehicle-to-infrastructure (V2I) [2]. 3GPP Rel. 16 defines the first standard for V2X based on a 5G new radio (NR) air interface [3].

Vehicles have limited computing and energy resources but need to process enormous amounts of data for advanced driving and automation. Offloading computationally intensive tasks to a cloud system can result in excessive propagation delays if vehicles are far from the centralized cloud. Network function virtualization (NFV) supports the flexible deployment of network and computing services closer to vehicles at locations such as roadside units (RSUs) and access network mobile edge computing (AN-MEC), which are ideal for latency-sensitive service processing [4]. NFV is instrumental in managing both the control and data planes of the network: the control plane functions as the network's brain, gathering information, abstracting network functionalities, and making critical decisions, such as offloading; the data plane then executes these decisions by forwarding packets from input interfaces to designated outgoing interfaces, effectively managing the dynamic conditions of modern networked systems [5].

In this paper, we examine a two-tier architecture comprising RSU and AN-MEC. In congested road intersections, often hotspots of traffic, vehicles frequently offload tasks to nearby RSUs to efficiently manage

* Corresponding author.
E-mail addresses: widhi.yahya@ub.ac.id (W. Yahya), ydlin@cs.nctu.edu.tw (Y.-D. Lin), marzuk@agh.edu.pl (F. Marzuk), cholda@agh.edu.pl (P. Chołda), laiyc@cs.ntust.edu.tw (Y.-C. Lai).

https://doi.org/10.1016/j.vehcom.2024.100862
Received 17 April 2024; Received in revised form 9 September 2024; Accepted 1 December 2024

Available online 5 December 2024

network load. To minimize computing latency, an overloaded RSU may offload tasks either horizontally to neighboring RSUs or vertically to AN-MEC. Rapid offloading decisions are critical in these hotspot traffic areas and are managed by the RSU's control plane, which determines both the offloading destination and volume, represented by the offloading ratio within a continuous action search space.

The ratio-based offloading within a continuous action search space highlights the limitations of traditional optimization methods, which involve extensive searches and often respond too slowly to dynamic traffic conditions, leading to outdated decisions. This is particularly problematic since the control plane's latency requirements are usually less stringent, typically in the range of milliseconds [6]. ML methods, particularly reinforcement learning (RL), provide a more effective solution by enabling suboptimal decisions in real time without the need for a fully optimal solution. RL is particularly advantageous in dynamic V2X environments where supervised learning is impractical due to the extensive and challenging data labeling required [7]. By learning directly from the environment, RL facilitates optimal offloading in V2X systems. It employs online learning and a trial-and-error phase, enabling the agent to quickly make initial decisions. These decisions are refined over time through continuous interaction with the environment, gradually approximating the optimal solution [8].

This study proposes an offloading strategy for a two-tier V2X system, allowing computing at both the RSU and the edge through vertical (RSU-to-AN-MEC) and horizontal (RSU-to-RSU and AN-MEC-to-AN-MEC) offloading. Offloading decisions, which consider vehicle resource availability and energy, typically occur at the application level. However, this study treats offloading as a control plane issue that requires rapid decision-making in response to fluctuating traffic, such as hotspot traffic. We employ the twin delayed deep deterministic policy gradient (TD3) algorithm, which addresses continuous action space challenges and demonstrates superior performance compared to the deep deterministic policy gradient (DDPG), trust region policy optimization (TRPO) and soft actor-critic (SAC) algorithms [9]. TD3 is used to perform ratio-based offloading, where the ratio determines both the destination and extent of offloading required. The main contributions and distinct features of this paper are summarized as follows:

- Introduction of a novel RL-based, ratio-based offloading optimization in a V2X system that efficiently manages both vertical and horizontal offloading. This optimization utilizes the TD3 algorithm within the network control plane, supported by a newly developed V2X testing environment that includes detailed latency models, state and action definitions, and a reward structure to assess offloading agent performance.
- Comprehensive comparative analysis of the TD3 with traditional optimization methods like simulated annealing (SA), focusing on key performance metrics such as decision and convergence times, decision quality, and the impacts of deploying single versus multiple offloading agents. This includes potential centralized deployment strategies to enhance coverage and operational efficiency.

The remainder of this paper is organized as follows. Section 2 presents the previous work on optimal ML-based offloading. Section 3 describes the V2X system, problem formulation, and latency models. Section 4 presents the solution algorithms, TD3 and SA, for ratio-based offloading. Section 5 describes the simulation environment and presents the numerical results. Section 6 concludes the paper.

2. Related work

This section reviews key approaches to offloading optimization in V2X systems, categorized into supervised learning, reinforcement learning, multi-agent reinforcement learning, advanced deep learning, and distributed optimization. We highlight the contributions, limitations, and differences from our work, see Table 1.

2.1. Supervised learning-based approaches

Supervised learning techniques have been employed for offloading optimization in V2X systems, where labeled data are used to train models. Sonmez et al. [10] introduced a two-stage supervised learning approach. In the first stage, they used a supervised machine learning algorithm to classify which destination can handle incoming tasks with either success or failure. The second stage applied regression to estimate the processing time required. Bo Fan et al. [19] optimized traffic offloading in vehicular networks with access points. Their goal was to maximize network throughput by adjusting offloading decisions. However, both approaches require substantial amounts of labeled data, which is expensive to collect in dynamic environments like V2X.

The dependency on labeled data in dynamic environments increases the cost and complexity of data collection, making it challenging for real-time and large-scale V2X systems.

2.2. Reinforcement learning (RL) approaches in V2X offloading

Reinforcement learning (RL) has emerged as a more flexible alternative to supervised learning in V2X systems, as it can learn the optimal offloading strategies by directly interacting with the environment. Zhou et al. [11] proposed a vehicular fog network (VFN) with three offloading modes: V2V, Vehicle-RSU-Vehicle (V2R2V), and Pedestrian-RSU-Vehicle (P2R2V). They utilized a multi-armed bandit (MAB)-based offloading approach, supported by supervised learning to predict vehicle mobility. They also incorporated coded computing to divide tasks into multiple subtasks, leveraging fog resources.

Ke et al. [12] optimized the binary offloading decision for subtasks by balancing the trade-offs between energy consumption and latency. They developed an adaptive offloading algorithm using the deep deterministic policy gradient (DDPG) method with Ornstein-Uhlenbeck noise, enhancing the exploration of action space. Shi et al. [13] addressed offloading and resource allocation in VFN, where vehicles could either process their tasks locally or offload them to nearby service vehicles. Their approach employed the soft actor-critic (SAC) algorithm to optimize resource utilization, making offloading decisions based on the available resources at the base station. Yang et al. [14] extended the offloading optimization to multi-hop vehicular networks, using DDPG to maximize computational throughput by gathering resources from multiple vehicles.

Khayyat et al. [17] applied deep Q-learning (DQN), using multiple deep neural networks (DNNs) to parallelize the offloading decision-making process. Zhang et al. [18] combined the analytic hierarchical process (AHP) with Q-learning to optimize both offloading and resource allocation while minimizing system costs. Li et al. [39] proposed a location-aware offloading mechanism using DDPG, further integrating task partitioning and scheduling strategies to reduce computational complexity.

These works focus on offloading decisions made by vehicles and user equipment (UE), leaving other components of the V2X system, such as RSUs, underexplored, particularly in terms of network-level offloading and handling hotspot traffic.

2.3. Multi-agent reinforcement learning (MARL) approaches

Multi-agent reinforcement learning (MARL) has been applied to more decentralized V2X systems, where multiple agents (vehicles, RSUs) cooperate to optimize offloading decisions. Zhu et al. [40] introduced an aerial relay station (ARS) for Internet of Vehicles (IoV) networks, minimizing system latency by using a multi-agent RL algorithm in a centralized training and distributed execution (CTDE) framework. In their binary offloading scenario, each vehicle made offloading decisions independently of other vehicles in the network. However, they did not account for queuing latency or realistic vehicular traffic.

Table 1
ML-based offloading optimization in V2X.

Paper	Agent Model									Network Model		
	Comparison with conventional optimization		Problem		Ratio-based	Online learning	Approach	Agent	Objective	Network target	Constraint	Unknown parameter
	Performance	Response time	Control plane	Mgmt. plane								
[10]	X	X	UE	X	X	O	ML supervised	1	Minimize latency	$V \uparrow E, V \uparrow C$	Resources	User mobility
[11]							MAB-based	1	Minimize latency	$V \leftrightarrow V, V \uparrow R \downarrow V, D \uparrow R \downarrow V$	Resources	User mobility
[12]							DDPG-based	1	Minimize cost	$V \uparrow E$	Resources	Available resources
[13]							SAC	1	Maximize utility	$V \leftrightarrow V$	Resources	Available resources
[14]							DDPG	1	Maximize throughput	$V \leftrightarrow V$	Resources	Available resources
[15]							CDRO DRL-based	1	Minimize computation completion time	$UE \uparrow E$	Resources	Available resources
[16]							DQN-based	1	Minimize latency and detection loss	$V \uparrow E, E \downarrow V$	Resources	Available resources
[17]				O			RL	1	Minimize latency and energy	$V \leftrightarrow V, V \uparrow R, V \uparrow I$	Resources	Available resources
[18]						X	AHP + Q learning	1	Minimize Cost and Energy	$V \leftrightarrow V, V \uparrow R$	Resources	Available resources
[19]			Network	X			DL	1	Maximize throughput	$V \leftrightarrow V, V \uparrow R$	Latency	Available resources
[20]						O	Quantum DRL-based	1	Minimize latency	$V \uparrow R, V \uparrow E$	Resources	Available resources
[21]							DDPG-based	1	Minimize latency and energy	$V \uparrow E, E \uparrow C$	Latency	Available resources
[22]			UE				DDPG-based	1	Minimize latency and energy	$V \uparrow R, R \uparrow E$	Latency and resources	Available resources
[23]				O			MADDPG-based	n	Minimize latency	$V \uparrow V, V \uparrow I, V \uparrow E$	Resources	Available resources
[24]			Network				DRL PPO	n	Minimize latency	$V \uparrow R, V \uparrow E$	Transmission power	Available resources
[25]					O		KKT-based	1	Minimize latency	$V \uparrow R$	Latency and resources	Available resources
[26]							DDPG-based PER	1	Minimize latency and energy	$V \uparrow E, V \uparrow C$	Resources	Available resources
[27]			UE	X			Actor-Critic	1	Minimize latency and energy	$V \uparrow E$	Latency and resources	Available resources
[28]							DDPG-based	1	Minimize latency	$V \uparrow E$	Resources	Available resources
[29]							MADDPG-based	n	Minimize latency and energy and rental price	$V \uparrow R, V \uparrow E$	Latency	Available resources
[30]					X		D4PG	n	Minimize latency and energy	$V \uparrow R, R \leftrightarrow R$	Latency and resources	Available resources
[31]							MARL C3O-based	n	Minimize latency	$V \leftrightarrow V, V \uparrow R$	Latency	Available resources
[32]							TD3-based DRL	n	Minimize latency and energy	$V \uparrow R, R \leftrightarrow R, R \downarrow V$	Resources	Available resources
[33]						X	MARL-based	n	Minimize latency	$V \leftrightarrow V, V \uparrow R, R \leftrightarrow R,$	Latency	Available resources
[34]						O	DQN-based	1	Maximize throughput	$V \uparrow E, E \leftrightarrow E$	Latency	Available resources
[35]							MADDPG-based	n	Minimize latency and energy	$V \leftrightarrow V, V \uparrow E$	Resources	Available resources
[36]					O		DQN-based	n	Minimize latency and energy	$V \uparrow E, E \downarrow V$	Resources	Available resources
[37]							Stackelberg-MADDPG	n	Minimize latency and task costs	$V \leftrightarrow V, V \uparrow I$	Resources	Available resources
[38]							MAB-based	1	Minimize latency	$V \leftrightarrow V$	Latency	Available resources
[39]	O			O	X		DDPG	1	Minimize latency and latency violation	$V \uparrow R, R \leftrightarrow R$	Resources	Available resources
[40]							MADRL	n	Minimize latency	$V \uparrow R, R \leftrightarrow R$	Latency	Available resources
[41]							MADDPG	n	Minimize latency and maximize reliability	$V \uparrow R, R \leftrightarrow R, R \downarrow V$	Resources	Available resources
[42]				X			MA-GAC	n	Minimize latency	$V \leftrightarrow V, V \uparrow R$	Resources	Arrival traffic
[43]						X	SMRL-MTO	n	Minimize latency	$V \uparrow E$	Resources	Arrival traffic
[44]				X		O	MADRL	n	Maximize utility	$V \leftrightarrow V, V \uparrow I$	Latency	Available resources
[45]			Network				MAB	n	Minimize latency	$V \uparrow R, V \uparrow E, R \uparrow E$	Resources	Available resources
[46]							Distributed-TD3	n	Minimize latency and energy	$E \leftrightarrow E, E \downarrow V$	Resources	Arrival traffic
Ours		O			O		MARL (TD3) vs. optimization algorithm	1,n*	Minimize latency	$V \uparrow R, R \leftrightarrow R, R \uparrow E$	Latency	Arrival traffic rate, Available resources

The used notation: O = Yes, X = No, C = Cloud, D = Pedestrian Device, E = Edge, I = Infrastructure, R = RSU, V = Vehicle.

Table 2
Notation definition for V2X with RSU architecture.

Category	Notations	Meaning	Attribute
Topology	$\mathcal{N}_i$	Set of RSU sites of each ith AN-MEC site, $\mathcal{N}_i = 1, 2, 3, ..., N$	Input
	$\mathcal{M}$	Set of AN-MEC site, $\mathcal{M} = 1, 2, 3, ..., M$	Input
	$\mathcal{H}_{i,j}$	Set of jth RSU's neighbors within ith AN-MEC	Input
	$d^{\mathrm{AA}}_{i_1 \leftrightarrow i_2}$	Distance between i_1th AN-MEC and i_2th AN-MEC site	Input
Capacity	μ^{A}_i	Capacity of ith AN-MEC site	Input
	$\mu^{\mathrm{R}}_{i,j}$	Capacity of jth RSU site of ith AN-MEC site	Input
	$B^{\mathrm{RA}}_{i,j}$	Uplink bandwidth of jth RSU site to ith AN-MEC site	Input
	$B^{\mathrm{AR}}_{i,j}$	Down bandwidth from ith AN-MEC to jth RSU site	Input
	$B^{\mathrm{RR}}_{i,j \leftrightarrow n}$	Bandwidth between jth RSU and mth RSU site	Input
	$B^{\mathrm{VR}}_{i,j}$	Uplink bandwidth at jth RSU of ith AN-MEC site	Input
	$B^{\mathrm{RV}}_{i,j}$	Downlink bandwidth at jth RSU of ith AN-MEC site	Input
Traffic	$\lambda_{i,j}$	Arrival task rate at jth RSU of ith AN-MEC site	Input
	$\lambda'_{i,j}$	Response rate at jth RSU of ith AN-MEC site	Input
	$\lambda^{\mathrm{R}}_{i,j}$	Traffic that is served at jth RSU site of ith AN-MEC	Output
	λ^{A}_i	Traffic that is served at ith AN-MEC site	Output
Delay	$l_{i,j}$	Average latency of arrival traffic at jth RSU of ith AN-MEC site	Input
	l	System average latency	Input
Offloading	$P^{\mathrm{R}}_{i,j}$	Probability of arrival traffic being served at jth RSU of ith AN-MEC site	Output
	$P^{\mathrm{RR}}_{i,j \rightarrow n}$	Probability of arrival traffic at jth RSU of ith AN-MEC site being offloaded to another nth RSU site at same AN-MEC	Output
	$P^{\mathrm{A}}_{i,j}$	Probability of arrival traffic at jth RSU of ith AN-MEC site being offloaded to another ith AN-MEC site	Output
	$P^{\mathrm{AA}}_{i \rightarrow m,j}$	Probability of arrival traffic at jth RSU of ith AN-MEC site being offloaded to another mth AN-MEC	Output

Cui et al. [41] proposed a collaborative edge computing scheme, formulating the joint optimization of offloading, computation, and delivery as a Markov decision process (MDP). They employed a multi-agent deep deterministic policy gradient (MADDPG) algorithm to solve this problem, although they did not address task uncertainties where only one RSU could handle an offloaded task.

Wei et al. [42] focused on a multi-tier task offloading framework in vehicular fog computing (VFC) systems. They designed a multi-agent gated actor-critic (MA-GAC) approach to optimize the offloading process in a distributed manner. Dai et al. [43] addressed multitask offloading (MTO) by designing a sequence-to-sequence (seq2seq)-based meta-reinforcement learning (meta-RL) algorithm. While their approach improved task offloading efficiency, it required enormous amounts of sampled data for training, leading to reduced sample efficiency. Hazarika et al. [44] designed a multi-agent RL algorithm to optimize task offloading and resource allocation in IoV systems. Their approach minimized task completion time by employing vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) offloading strategies.

Generally, MARL approaches suffer from high computational complexity, resulting in slower execution times and limiting their effectiveness for real-time V2X applications.

2.4. Advanced RL and DRL-based approaches

In more recent work, advanced RL and deep reinforcement learning (DRL) techniques have been proposed to optimize offloading in V2X systems. Qiu et al. [21] formulated an optimization problem to minimize total task latency and energy consumption in VEC systems, proposing a DDPG-based adaptive computation offloading and power allocation algorithm (DDPG-ACOPA). Wu et al. [22] applied DDPG with a multi-head self-attention mechanism to minimize latency and energy consumption in multi-UAV multi-server MEC systems.

Hu et al. [23] explored integrated sensing and communication (ISAC) in V2X MEC networks, optimizing task offloading and resource allocation with a MADDPG-based approach. Cong et al. [24] used a proximal policy optimization (PPO) algorithm in multi-user multi-server vehicular networks to reduce system delay, complemented by a power allocation algorithm for delay minimization. While these approaches offer improved offloading strategies, none of them address ratio-based or horizontal task offloading mechanisms.

These studies do not incorporate ratio-based or horizontal offloading, limiting their ability to efficiently balance loads between multiple RSUs or edge servers.

2.5. Optimization in distributed and cooperative systems

Several recent studies have focused on optimization of distributed task offloading in V2X environments. Wang et al. [27] developed a DQN-based algorithm to optimize partial offloading, while Chen et al. [31] designed a competitive and cooperative computation offloading model (C3O) to minimize task processing delays in VEC networks. Their multi-agent RL-based solution enables vehicles to cooperatively decide the optimal offloading policy based on network conditions.

Fan et al. [32] proposed a resource allocation scheme in VEC IoV scenarios, using a TD3-based DRL algorithm to minimize processing latency and energy consumption. Hou et al. [33] introduced a hierarchical VFC system to optimize computational resource allocation, applying a counterfactual multi-agent RL algorithm. Ning et al. [34] considered joint task offloading and service migration in RIS-assisted VEC systems, proposing a DQN-based algorithm to maximize network throughput. Qin et al. [35] tackled the problem of air-ground vehicular cooperation by combining RSUs and UAVs to minimize delay and energy consumption using greedy and MADDPG algorithms.

These works overlook horizontal task offloading, which could further improve resource utilization and reduce latency by distributing tasks across multiple RSUs and edge servers.

This study introduces a novel approach that enables both horizontal and vertical ratio-based offloading within a continuous action space. We utilize the TD3 algorithm to effectively manage this offloading, focusing on optimizing task distribution to enhance response times and reduce the average delay encountered by arriving tasks.

3. V2X with RSU: system and problem formulation

This section describes the V2X architecture, problem formulation, and latency models. Table 2 provides notations used for problem formulation and latency models.

3.1. V2X with RSU architecture

V2X communication is a key service within the 5G network ecosystem, where vehicles are equipped with onboard units (OBUs) to exe-

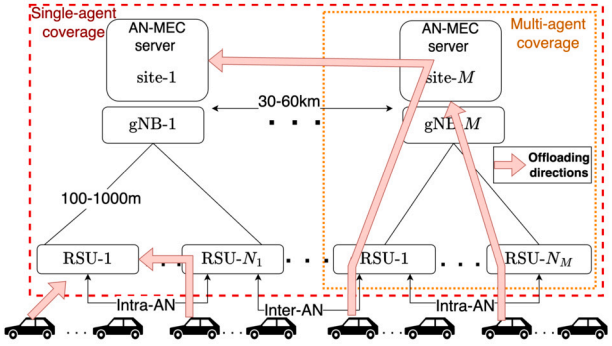


Fig. 1. V2X with RSU architecture.

cute various applications. These OBUs communicate with other vehicles (V2V), infrastructure such as RSUs, and networks (V2N) including base stations [47]. Both RSUs and base stations, referred to in this study as RSUs and AN-MECs respectively, are equipped with computing resources, enabling task offloading between them to avoid overloading any single computing site. This study considers a two-tier computing structure comprising RSUs and AN-MECs. In the upper tier, AN-MEC servers are placed in 5G base stations, which typically cover up to 30 km, with the distance between base stations reaching up to 60 km [48]. In the bottom tier of the V2X system, vehicles linked to RSUs generate numerous tasks via sensors and applications.

The network control plane functions as an offloading agent, managing the distribution of tasks to prevent overload. NFV enables hosting the offloading agent either in a CO, which connects several base stations with AN-MEC, or collocated with a base station [49]. The first placement represents a V2X system with a single offloading agent, while the latter represents a V2X system with multiple agents, each covering a smaller area, as shown in Fig. 1. In this study, the agent employs the TD3 algorithm, using environment information, including computing and communication capacities, to determine the optimal offloading ratio that minimizes the average latency experienced by vehicles.

Vehicles can vertically offload tasks to a current-connected RSU to minimize computing latency. This offloading process is managed by vehicle applications, considering their individual computing and energy resources, and is beyond the scope of the network control plane problem addressed in this paper. When an RSU becomes overloaded, it can offload tasks horizontally to its directly connected neighboring RSUs or vertically to an AN-MEC. Similarly, an overloaded AN-MEC can offload tasks horizontally to other AN-MECs. These offloading directions are illustrated in Fig. 1.

3.2. Latency model

In the V2X-RSU system, traffic computation is offloaded to RSUs and AN-MECs to minimize latency and power consumption while utilizing abundant computing resources. RSUs extend edge functionalities to vehicles, while AN-MECs augment RSU capabilities to handle high traffic volumes. The subsequent subsections detail four traffic directions.

1. RSU traffic: The RSU is the initial entity responsible for processing tasks upon vehicle arrival, encompassing RSU communication and computing latencies. The communication latency of an RSU is influenced by the uplink and downlink bandwidths, denoted by $B^{\mathrm{VR}}_{i,j}$ and $B^{\mathrm{RV}}_{i,j}$ respectively. The likelihood of traffic being processed by a receiving RSU is denoted as $P^{\mathrm{R}}_{i,j}$. If the number of generated tasks surpasses the RSU's computing capacity, tasks can be offloaded to external computing resources at RSU neighbors or AN-MEC sites. The latency experienced by traffic served on the jth RSU connected to the ith AN-MEC is expressed as

$$l^{\mathrm{R}}_{i,j} = \frac{1}{B^{\mathrm{VR}}_{i,j} - \lambda_{i,j}} + \frac{1}{\mu^{\mathrm{R}}_{i,j} - \lambda^{\mathrm{R}}_{i,j}} + \frac{1}{B^{\mathrm{RV}}_{i,j} - \lambda'_{i,j}}, \tag{1}$$

where

$$\lambda^{\mathrm{R}}_{i,j} = P^{\mathrm{R}}_{i,j}\lambda_{i,j} + \sum_{n\in\mathcal{H}_{i,j}} P^{\mathrm{RR}}_{i,n\to j}\lambda_{i,n}.$$

The aggregate traffic reaching the jth RSU, denoted as $\lambda^{\mathrm{R}}_{i,j}$, includes both vertically offloaded traffic from connected vehicles and horizontally offloaded traffic from adjacent RSUs.

2. RSU neighbors' traffic: when the receiving RSU becomes overloaded, with a probability of $P^{\mathrm{RR}}_{i,j\to n}$, tasks are offloaded to RSU neighbors. This traffic involves communication latency at the RSU, computing latency at the nth RSU, and communication latency between the jth and nth RSUs. RSUs are interconnected via a gigabit wired link with maximum bandwidth $B^{RR}_{i,j\leftrightarrow n}$. The latency experienced by arrival traffic served at the nth RSU neighbor is formulated as

$$l^{\mathrm{RR}}_{i,j\to n} = \frac{1}{B^{\mathrm{VR}}_{i,j} - \lambda_{i,j}} + \frac{1}{B^{\mathrm{RR}}_{i,j\leftrightarrow n} - P^{\mathrm{RR}}_{i,j\to n}\lambda_{i,j}} + \frac{1}{\mu^{\mathrm{R}}_{i,n} - \lambda^{\mathrm{R}}_{i,n}} + \frac{1}{B^{\mathrm{RR}}_{i,j\leftrightarrow n} - P^{\mathrm{RR}}_{i,j\to n}\lambda'_{i,j}} + \frac{1}{B^{\mathrm{VR}}_{i,j} - \lambda'_{i,j}}, \tag{2}$$

where

$$\lambda^{\mathrm{R}}_{i,n} = P^{\mathrm{R}}_{i,n}\lambda_{i,n} + \sum_{x\in\mathcal{H}_{i,n}} P^{\mathrm{RR}}_{i,x\to n}\lambda_{i,x}.$$

The total traffic arriving at the nth RSU, $\lambda^{\mathrm{R}}_{i,n}$, is composed of vertically offloaded traffic from connected vehicles and horizontal offloaded traffic from vehicles in neighboring RSUs.

3. AN-MEC Traffic: hotspot traffic may overload some of the RSU sites, and some of the traffic can be offloaded to AN-MEC. Traffic from jth RSU of ith AN-MEC has the probability of $P^{\mathrm{A}}_{i,j}$ being offloaded to the directly connected ith AN-MEC. Offloaded traffic latency to the directly connected ith AN-MEC site is calculated as RSU uplink-downlink latency + uplink-downlink latency between the jth RSU and the ith AN-MEC + ith AN-MEC computing latency and is expressed as

$$l^{\mathrm{A}}_{i,j} = \frac{1}{B^{\mathrm{VR}}_{i,j} - \lambda_{i,j}} + \frac{1}{B^{\mathrm{RA}}_{i,j} - \left(P^{\mathrm{A}}_{i,j}\lambda_{i,j} + \sum_m^M P^{\mathrm{AA}}_{i\to m,j}\lambda_{i,j}\right)} + \frac{1}{\mu^{\mathrm{A}}_{i} - \lambda^{\mathrm{A}}_{i}} + \frac{1}{B^{\mathrm{AR}}_{i,j} - \left(P^{\mathrm{A}}_{i,j}\lambda'_{i,j} + \sum_m^M P^{\mathrm{AA}}_{i\to m,j}\lambda'_{i,j}\right)} + \frac{1}{B^{\mathrm{RV}}_{i,j} - \lambda'_{i,j}}, \tag{3}$$

where

$$\lambda^{\mathrm{A}}_{i} = \sum_{j\in\mathcal{N}_i} P^{A}_{i,j}\lambda_{i,j} + \sum_{m\in\mathcal{M}\setminus\{i\}}\sum_{j\in\mathcal{N}_m} P^{AA}_{m\to i,j}\lambda_{m,j}.$$

The total traffic arriving at the ith AN-MEC, λ^{A}_{i}, is comprised of the offloaded traffic from its RSU and the horizontal offloaded traffic from the RSUs of neighboring AN-MECs.

4. AN-MEC neighboring traffic: Hotspot traffic may overload all RSUs of the ith AN-MEC as well as the ith AN-MEC site. Some of the arrival traffic could be offloaded to AN-MEC neighbors with probability $P^{\mathrm{AA}}_{i\to m,j}$. The differences in latency composition with AN-MEC traffic, this traffic also experiences propagation latency between ith AN-MEC site and its neighbor, mth AN-MEC site. So, the latency experienced by the traffic served at the mth AN-MEC site is formulated as

$$l^{\mathrm{AA}}_{i\to m,j} = \frac{1}{B^{\mathrm{VR}}_{i,j} - \lambda_{i,j}} + \frac{1}{B^{\mathrm{RA}}_{i,j} - \left(P^{\mathrm{A}}_{i,j}\lambda_{i,j} + \sum_m^M P^{\mathrm{AA}}_{i\to m,j}\lambda_{i,j}\right)} +$$

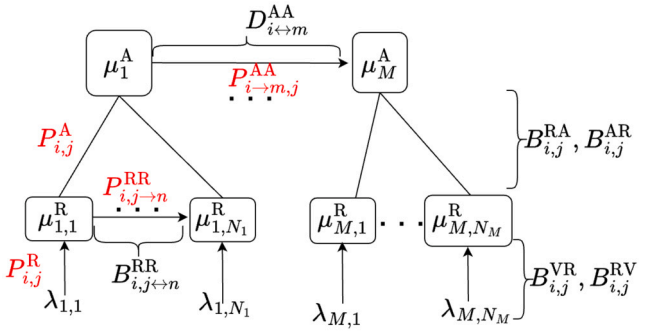


Fig. 2. V2X with RSU notations.

$$\frac{1}{\mu^{A}_{m}-\lambda^{A}_{m}}+\frac{1}{B^{AR}_{i,j}-\left(P^{A}_{i,j}\lambda'_{i,j}+\sum_{m}^{M}P^{AA}_{i\to m,j}\lambda'_{i,j}\right)}+\frac{1}{B^{RV}_{i,j}-\lambda'_{i,j}}+2\times D^{AA}_{i\leftrightarrow m}, \tag{4}$$

where

$$\lambda^{A}_{m}=\sum_{j\in\mathcal{N}_m}P^{A}_{m,j}\lambda_{m,j}+\sum_{i\in\mathcal{M}\setminus\{m\}}\sum_{j\in\mathcal{N}_m}P^{AA}_{i\to m,j}\lambda_{m,j}.$$

5. Average V2X with RSU system latency: The average latency of arrival traffic at the jth RSU of the ith AN-MEC can be determined by equations (1), (2), (3), and (4), and weighted by the arrival traffic at each site, which is expressed as

$$l_{i,j}=\frac{l^{R}_{i,j}\left(P^{R}_{i,j}\lambda_{i,j}\right)+\sum_{n\in\mathcal{N}_i\setminus\{j\}}l^{RR}_{i,j\to n}\left(P^{RR}_{i,j\to n}\lambda_{i,j}\right)}{\lambda_{i,j}}+\frac{l^{A}_{i,j}\left(P^{A}_{i,j}\lambda_{i,j}\right)+\sum_{m\in\mathcal{M}\setminus\{i\}}l^{AA}_{i\to m,j}\left(P^{AA}_{i\to m,j}\lambda_{i,j}\right)}{\lambda_{i,j}}. \tag{5}$$

Finally, the system's average latency is calculated by

$$l=\sum_{i}\sum_{j}\frac{\lambda_{i,j}\times l_{i,j}}{\sum_{i}\sum_{j}\lambda_{i,j}}. \tag{6}$$

3.3. V2X with RSU problem statement

Given some incoming traffic rates, including hotspot traffic in an RSU, the control plane of the V2X system determines the best offloading ratio. The offloading ratio for each arriving traffic site is depicted in Fig. 2 which comprises traffic served, RSU, RSU neighbors, AN-MEC, and AN-MEC neighbors. The objective again is to minimize the average latency of the arrival traffic. We define the problem as follows: input: topology parameters are $\mathcal{M},\mathcal{N}_i$, and $\mathcal{H}_{i,j}$; distance between entities $d^{AA}_{i_1\leftrightarrow i_2}$; capacity of the system includes μ^{A}_{i}, $\mu^{R}_{i,j}$, $B^{RA}_{i,j}$, $B^{AR}_{i,j}$, $B^{RR}_{i,j\leftrightarrow n}$, $B^{VR}_{i,j}$, and $B^{RV}_{i,j}$; arrival and response traffic $\lambda_{i,j},\lambda'_{i,j}$; current sites and system average latency $l_{i,j},l$. The output is the offloading ratio $P^{R}_{i,j}$, $P^{RR}_{i,j\to n}$, $P^{AA}_{i,j}$, and $P^{AA}_{i\to m,j}$.

For the problem described, the V2X architecture consists of two tiers of computing resources: RSU and AN-MEC. Each of the RSU and AN-MEC sites is equipped with a number of servers with a total capacity of $\mu^{R}_{i,j}$, μ^{A}_{i}. The vehicles are connected to the RSU via a wireless link with capacity for uplink and downlink communications, denoted by $B^{VR}_{i,j}$, and $B^{RV}_{i,j}$. $\mathcal{N}_i$ denotes a number of RSUs connected to the ith AN-MEC site. RSUs are interconnected through gigabit links, the capacity of which is represented by $B^{RR}_{j\leftrightarrow n}$. Because $B^{RR}_{j\leftrightarrow n}$ is limited, we limit the horizontal offloading of RSUs to adjacent RSUs within the same AN-MEC. $\mathcal{H}_{i,j}$ is a set of jth RSU's neighbors connected to ith AN-MEC site. The uplink and downlink capacities of a communication link between RSU and AN-MEC are indicated by $B^{RA}_{i,j}$, $B^{AR}_{i,j}$. The propagation latency is not considered between RSUs and between RSU and AN-MEC because the distance between them is short, between one hundred meters and a unit of a kilometer. The propagation latency is considered between the AN-MEC sites because the distance between them $d^{AA}_{i_1\leftrightarrow i_2}$ is large, approximately 10 to 30 km. We ignored the transmission latency between the AN-MECs because the links employ fiber optic with terabit capacity, which is large enough compared to the arrival traffic.

Arrival traffic, $\lambda_{i,j}$, is generated by vehicles connected to the jth RSU site and the ith AN-MEC site. Arrival traffic can be handled locally at the arrival RSU site with the probability $P^{R}_{i,j}$. An overloaded RSU could offload some traffic horizontally to its neighbors with probability $P^{RR}_{i,j\to n}$, vertically to a directly linked AN-MEC with probability $P^{A}_{i,j}$, and vertically to neighboring AN-MEC sites with probability $P^{AA}_{i\to m,j}$. V2X employs single or multiple agents learning in a CO or AN-MEC, respectively. The learning agent, which is part of the control plane, uses SA and TD3 algorithms to determine the optimal offloading decision to minimize the average system latency l.

4. Ratio-based offloading with TD3 and SA

4.1. TD3-based offloading

Algorithm 1: TD3 [9].

1 Initialize critic network $Q_{\theta_1}, Q_{\theta_2}$ and actor network π_ϕ with random parameters θ_1,θ_2,ϕ_i;
2 Initialize target networks, $\theta'_1\leftarrow\theta_1,\theta'_2\leftarrow\theta_2,\phi'\leftarrow\phi$;
3 Initialize replay buffer $\mathcal{B}$;
4 **for** $t=1$ *to* T **do**
5 select random action $a_i\sim\pi_\phi(s)+\epsilon$, $\epsilon\sim\mathcal{N}(0,\sigma)$ and observe reward r and new state s' ;
6 Store transition tuple (s,a,r,s') in $\mathcal{B}$;
7 Sample mini-batch of N transition (s,a,r,s') from $\mathcal{B}$;
8 $\tilde{a}\leftarrow\pi_{\phi'}(s')+\epsilon$, $\epsilon\sim clip(\mathcal{N}(0,\tilde{\sigma}),-c,c)$;
9 $y\leftarrow r+\gamma\left(\min_{i=1,2}Q_{\phi_i}\left(s',\tilde{a}\right)\right)$;
10 Update critics $\theta_i\leftarrow\text{argmin}_{\theta_i}\,N^{-1}\sum\left(y-Q_{\theta_i}(s,a)\right)^2$;
11 **if** t *mod* d **then**
12 Update actor parameter ϕ policy gradient: $\nabla_\phi J(\phi)=N^{-1}\sum\nabla_a Q_{\theta_1}(s,a)\big|_{a=\pi_\phi(s)}\nabla_\phi\pi_\phi(s)$;
13 Update target network:
14 $\theta'_i\leftarrow\tau\theta_i+(1-\tau)\theta'_i$;
15 $\phi'\leftarrow\tau\phi+(1-\tau)\phi'$;
16 **end**
17 **end**

Fig. 3 illustrates the TD3 structure, consisting of three primary elements: environment, agent, and replay buffer. The offloading agent utilizes system information states s, such as computing and communication capacity, to determine the optimal offloading decision a, resulting in a high reward r (indicative of low latency), thus altering the system condition to s'. The interactions history (s,a,s',r) is stored in the replay buffer $\mathcal{B}$ for the training process outlined in lines 4 to 17 of Algorithm 1.

The training process in the TD3 algorithm initiates with the actor network π_ϕ taking the system state s to produce a random action a, thereby transitioning the system state to s'. These transactions are saved in $\mathcal{B}$ until a sufficient number accumulates. The transition batches $\mathcal{N}$ are then sampled from $\mathcal{B}$, and the next action $\tilde{a}$ is predicted by feeding s' into the actor target network $\pi_{\phi'}$ and augmenting it with exploration noise ϵ to enhance learning stability. TD3 utilizes twin critic target networks to calculate Q-values, which are derived from the minimum of Q-values of twin critic targets multiplied by the discount factor γ and augmented by the current reward r to mitigate overestimation.

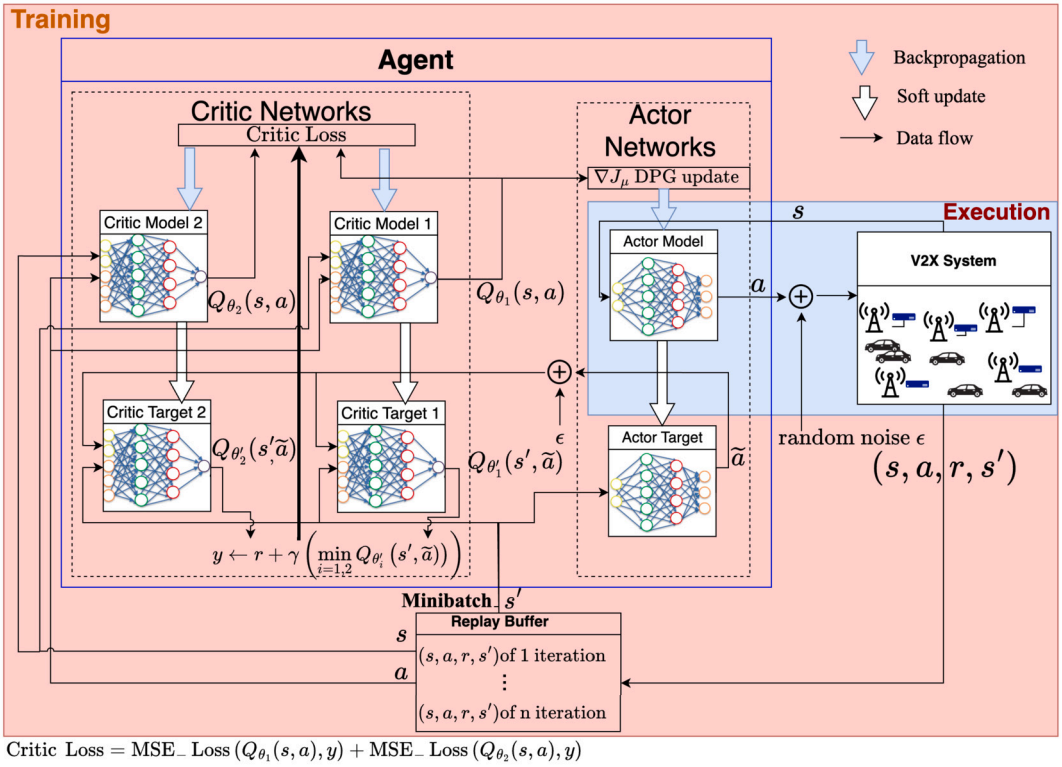


Fig. 3. TD3-based offloading structure.

The target value y is calculated as the sum of the current reward r and the minimum of the Q-values of the twin-critic targets, scaled by the discount factor γ. Mathematically, it can be expressed as

$$y \leftarrow r+\gamma\left(\min_{i=1,2} Q_{\phi_i}\left(s',\tilde{a}\right)\right). \tag{7}$$

This target value y serves as a reference for updating the critic models. The critic loss is then computed as the mean squared error (MSE) between the target y and the Q-values predicted by the twin-critic models $Q_{\theta_i}(s,a)$, facilitating the update of the critic models' parameters θ_i using gradient descent. To maintain stability, the actor and twin-critic target networks are updated every d iteration through soft updates, adjusting their parameters (ϕ, θ_1, θ_2) by a fraction τ of their model weights. In particular, only a fraction of the parameters are updated at each iteration. Finally, the actor model is updated through the deterministic policy gradient method.

The subsequent subsections formulate the contexts of s, a, r within the framework of V2X with RSU systems.

4.1.1. State

The offloading agent can be deployed either at a centralized cloud/CO linked to several base stations with AN-MECs or within each AN-MEC connected to multiple RSUs, forming a multi-agent system. This paper investigates the impact of single-agent and multi-agent optimization on offloading. The distinction lies in the coverage area managed by the agent: a single-agent covers the entire V2X systems, accommodating large input and action sizes, whereas multiple agents reduce coverage, resulting in smaller input and action sizes.

The learning agent is trained using prior experiences. The memory buffer (memory) stores experiences in the form of state, action, next-state, and reward (s,a,s',r). The agent will learn what is the best-action (offloading ratio) that should be taken given a particular environment's state. The state represents the current system's status, which is often determined by monitoring system parameters such as traffic arrival rate, communication capacity, computation capacity, and system latency. The action will transition a state from s to s' with larger or smaller latency. The optimal action minimizes the average system latency and is rewarded positively.

Table 3
Observable states of single and multi offloading agents.

Input	Single-agent	Multi-agent
T	$\left\{\lambda_{i,j}\right\}_{i=1,j=1}^{M,N_i}, \left\{\lambda'_{i,j}\right\}_{i=1,j=1}^{M,N_i}$	$\left\{\lambda_{i,j}\right\}_{j=1}^{N_i}, \left\{\lambda'_{i,j}\right\}_{j=1}^{N_i}$
C	$\left\{\mu_i^{\mathrm{A}}\right\}_{i=1}^{M}, \left\{\mu_{i,j}^{\mathrm{R}}\right\}_{i=1,j=1}^{M,N_i}$	$\mu_i^{\mathrm{A}}, \left\{\mu_{i,j}^{\mathrm{R}}\right\}_{j=1}^{N_i}$
N	$\left\{B_{i,j\leftrightarrow n}^{\mathrm{RR}}, B_{i,j}^{\mathrm{RA}}\right\}_{i=1,j=1,}^{M,N_i}$ $\left\{B_{i,j}^{\mathrm{AR}}, B_{i,j}^{\mathrm{VR}}, B_{i,j}^{\mathrm{RV}}\right\}_{i=1,j=1,}^{M,N_i}$	$\left\{B_{i,j\leftrightarrow n}^{\mathrm{RR}} B_{i,j}^{\mathrm{RA}}\right\}_{j=1}^{N_i}$ $\left\{B_{i,j}^{\mathrm{AR}}, B_{i,j}^{\mathrm{VR}}, B_{i,j}^{\mathrm{RV}}\right\}_{j=1,}^{N_i}$
L	$\left\{l_{i,j}\right\}_{i=1,j=1}^{M,N_i}$	$\left\{l_{i,j}\right\}_{j=1}^{N_i}$

The following states constitute the V2X system information: arrival and response traffic rate ($\lambda_{i,j}, \lambda'_{i,j}$), computing resources ($\mu_{i,j}^{\mathrm{R}}, \mu_i^{\mathrm{A}}$), communication resources ($B_{i,j}^{\mathrm{VR}}$, $B_{i,j}^{\mathrm{RV}}$, $B_{i,j\leftrightarrow n}^{\mathrm{RR}}$, $B_{i,j}^{\mathrm{AR}}$, $B_{i,j}^{\mathrm{RA}}$), system latency ($l_{i,j}$). State arrays are constructed from the states and can be used as inputs to policy and Q-value networks.

Table 3 presents a comparison of observable system states between single and multi-offloading agents in the V2X with RSU system. T denotes a set of arrival and response traffic rates for all vehicles. C represents arrays that contain information on computing capabilities in vehicles, RSU sites, and AN-MECs. Similarly, N is an array that contains information on the capacity of communication links, which varies over time in response to traffic arrival and response rates. L denotes the average latency across all sites. T, C, N, and L are combined to form the environment state,

$$s = [T, C, N, L]. \tag{8}$$

4.1.2. Action

Action represents the offloading ratio at V2X with RSU which derives some probability of offloaded traffic destinations formulated as local RSU ($P_{i,j}^{\mathrm{R}}$), RSU neighbors ($P_{i,j\rightarrow n}^{\mathrm{RR}}$), AN-MEC ($P_{i,j}^{\mathrm{A}}$), and AN-MEC neighbors ($P_{i\rightarrow m,j}^{\mathrm{AA}}$). The total load ratio of each arrival traffic rate at

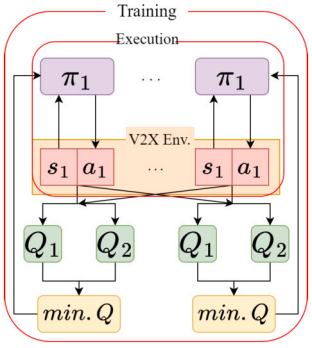


Fig. 4. Multi-agent TD3 structure.

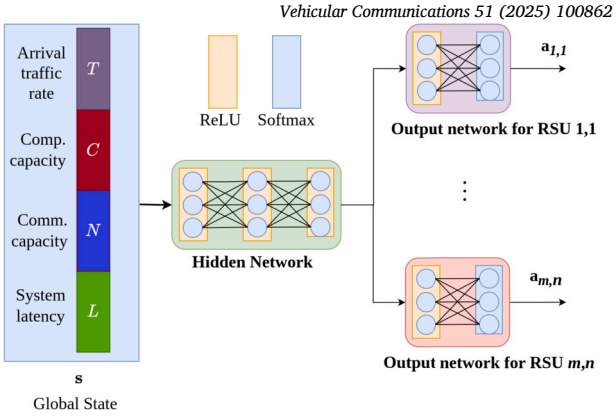


Fig. 5. Ratio-based offloading actor model.

jth RSU of ith AN-MEC site is one, $P^{\mathrm{R}}_{i,j} + \sum_{n\in\mathcal{N}_i\setminus\{j\}} P^{\mathrm{RR}}_{i,j\to n} + P^{\mathrm{A}}_{i,j} + \sum_{m\in\mathcal{M}\setminus\{i\}} P^{\mathrm{AA}}_{i\to m,j} = 1$. The action set, which is a product of the actor's network, is denoted by

$$a = \left\{ P^{\mathrm{R}}_{i,j}, P^{\mathrm{RR}}_{i,j\to n}, P^{\mathrm{A}}_{i,j}, P^{\mathrm{AA}}_{i\to m,j}, \forall n \in \mathcal{N}_i\setminus\{j\}, \forall m \in \mathcal{M}\setminus\{i\} \right\}_{i=1,j=1}^{M,N_i} \tag{9}$$

for single-agent TD3 which the agent is hosted in a centralized CO connected to some AN-MEC sites. In multi-agent TD3, each agent is hosted at an AN-MEC site and is only responsible for offloading decisions that belong to the ith AN-MEC, which includes all RSU sites connected to it,

$$a = \left\{ P^{\mathrm{R}}_{i,j}; P^{\mathrm{RR}}_{i,j\to n}; P^{\mathrm{A}}_{i,j}; P^{\mathrm{AA}}_{i\to m,j}, \forall n \in \mathcal{N}_i\setminus\{j\}, \forall m \in \mathcal{M}\setminus\{i\} \right\}_{j=1}^{N_i}. \tag{10}$$

4.1.3. Reward

We do not train the agent by creating a best-action direction using labeled data, but rather by generating a reward signal to represent the quality of the action taken. A positive reward is assigned to actions that contribute to achieving the system's objectives, while actions that do not are penalized with a negative reward. The objective of RL is to maximize this reward. Given that the goal of offloading optimization is to minimize average latency, the reward for an action taken at time t is defined as

$$r = \frac{1}{l}, \tag{11}$$

where l is derived from equation (6).

4.1.4. Multi-agent structure

The agent utilizes Neural Networks (NN) for actor models, actor targets, twin critic models, and twin critic targets, interacting with the environment during execution to determine the offloading ratio using the actor model. The history of the interaction is stored in the replay buffer as tuples of state, action, subsequent state, and reward (s_t, a_t, s', r). TD3 combines value and policy approaches, optimizing actor model parameters for the best offloading decision and producing Q-values for expected returns with critic models. Twin-critic models prevent Q-value overestimation, updating policy parameters using gradient descent. The actor and critic targets remain constant for some iterations for stable learning. Additionally, this study compares single-agent and multi-agent RL, evaluating large systems with a single agent and small systems with multiple agents, with differences in the observation state and action space each agent collects.

In the execution time, as depicted in Fig. 4, the policy network of a multi-agent RL (π) observes the environment locally to determine a local offloading ratio (offloading ratio of an MEC site). All offloading agents' policies and Q-value networks will be updated via a training procedure incorporating global system data. This global information consists of the previous data of all agents, which is stored in a replay buffer. An offloading agent must determine the offloading ratios for all MEC sites after observing the global environment state in single-agent RL. This requires a large input space from the global observation and a large action space for all MEC sites' offloading ratios. Like multi-agent RL, single-agent policy and Q-value networks are trained using previous data collected in a replay buffer.

To minimize the system latency l, V2X with RSU offloading agent determines the optimal offloading decision. The objective is then expressed as a reward function of the RL environment, as illustrated by equation (11). The TD3 algorithm employed by the offloading agent maximizes the system's reward.

As illustrated in Fig. 3, each V2X agent with RSU offloading uses six NNs for two actor networks (model and target) and four critic networks (twin models and twin targets). The actor model is responsible for determining the optimal offloading ratio. This ratio includes both the destination of the offloaded traffic (vertical and horizontal directions) and the amount of traffic that will be offloaded. The offloading ratio in a single-agent environment is expressed by equation (9). While the offloading ratio for multiple agents is given by equation (10).

Since ratio-based offloading is considered in V2X with RSU, the Softmax activation function is used for policy or actor networks. The output of the Softmax function is a categorical distribution that classifies the input. In our case, the class denotes the destination for offloading. While a class's confidence ratio represents the offloading ratio to an RSU/AN-MEC site, Fig. 5 depicts the structure of actor networks for single-agent and multi-agent RL. In the execution phase, the single-agent actor network uses the global state, as defined by equation (8), as input to determine the global action, which consists of the offloading ratio of all offloading destinations. To compute the local offloading ratio at -MEC site, a multi-agent actor network utilizes local state information about entities that may become the offloading destination for the jth RSU task. Because they are not the offloading destination, an agent does not need to maintain information on another RSU belonging to another AN-MEC. The input and output space of the multi-agent actor's network is smaller than that of the single-agent actor's network.

4.2. SA-based offloading

SA is a global optimization technique inspired by metallurgical annealing. SA avoids local optima that may occur in the Genetic and Particle-Swarm algorithms by allowing for certain, less desirable solutions in a controlled manner via probability [50].

Algorithm 2 dictates the decision-making process of the offloading agent in each RSU, optimizing the offloading ratio based on the initial observable states, such as system state/information (s). The algorithm begins by defining an initial action (a) and calculating the corresponding reward (r), to minimize the average latency experienced by UEs.

During each iteration, a new action (a_{new}) is selected. This selection involves transferring a random fraction of tasks, multiplied by a predetermined step size (α), from the source RSU site to a randomly chosen MEC site. Subsequently, the performance of the new solution (r_{new}) is

Algorithm 2: Simulated Annealing.

Input: system observable states (s)
Result: optimized offloading ratio a
Initialize action: randomly generate offloading ratio a;
Initialize reward: $r = f(s, a)$;
Initialize parameters: temperature T, Boltzmann's constant k, reduction factor c;
for *each AN-MEC site (M)* **do**
  for *each RSU site (N_i)* **do**
    for *each temperature iteration (R)* **do**
      for *each neighbor to visit (S)* **do**
        Select a new action $a_{new} = f(a, \alpha)$;
        $r_{new} = f(s, a_{new})$;
        if $r_{new} > r$ **then**
          $r = r_{new}$;
          $a = a_{new} anAN$;
        else
          $\Delta r = r_{new} - r$;
          $rand \leftarrow \text{random}(0, 1)$;
          if $rand < \frac{1}{e^{\left(\frac{\Delta r}{kT}\right)}}$ **then**
            $r = r_{new}$;
            $a = a_{new}$;
          else
            $r = r$;
            $a = a$;
          end
        end
      end
      Decrease the temperature periodically: $T = c \times T$;
    end
  end
end

compared to that of the previous solution (r), with $\Delta r = r_{new} - r$. If the previous solution proves superior, it is retained; otherwise, the new solution is selected with a probability determined by equation (12):

$$\frac{1}{e^{\left(\frac{\Delta r}{kT}\right)}} \tag{12}$$

where k and T denote Boltzmann's constant and temperature, respectively.

If the result of equation (12) exceeds a random float between (0,1), a poor decision can be made. Otherwise, the previous solution is preserved if it is superior to the new one. SA is more likely to accept a suboptimal solution at high temperatures and is less likely to accept a suboptimal solution at low temperatures [51].

5. Simulations and results

5.1. Simulation setup and experimental settings

V2X with RSU is a system with hundreds to thousands of computing and communication resources. The cost of constructing a testbed for offloading optimization research is expensive. Since the primary purpose of the proposal is to compare traditional optimization with RL-based offloading techniques, simulation may be a more cost-effective option to create a testbed. In addition, the real network control plane utilizes network abstractions comparable to simulation models and optimization techniques based on these abstractions.

The simulation environment is written in Python and runs on a Core i7 10th gen processor with 16 GB RAM and an RTX2060s GPU. Table 4 details the simulation parameters, classified as topology, RAN and site capacities, and traffic. The topology settings are designed to approximate real system implementations, considering factors such as the distances between entities, and computing and communication capacities. Entities are distributed in an area using a geographic coordinate sampling application (https://epitools.ausvet.com.au/rgcs). Link capacity in packets/s is calculated by dividing the bandwidth by p. Ten experiments were conducted for each scenario and algorithm. Each TD3 iteration comprised 10,000 episodes with 10 steps each, representing various states of system utilization. The SA iteration depends on the number of AN-MECs, RSUs, temperatures M, and neighbors visited N.

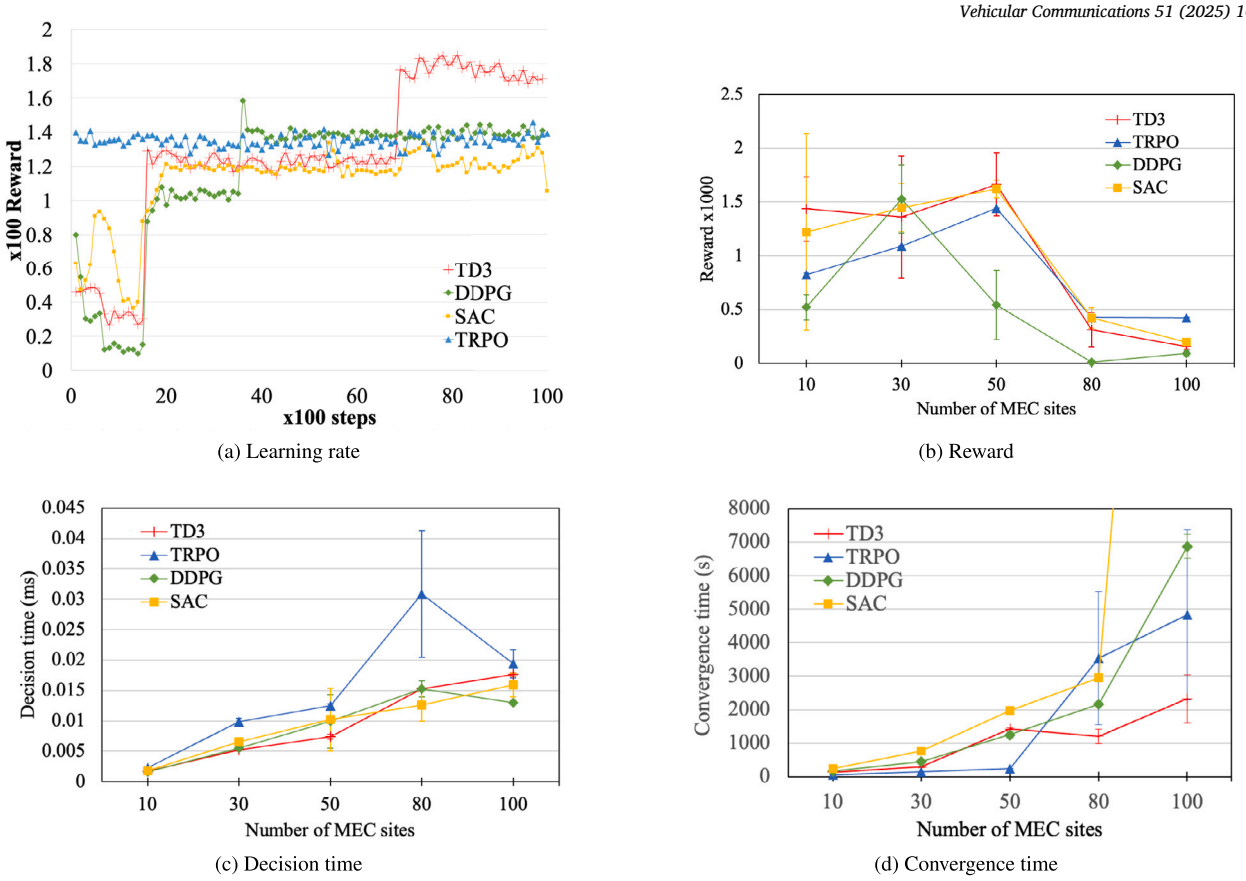
Table 5 and Table 6 detail the settings for the SA and TD3 algorithms, with Optuna used to optimize these settings [52]. Offloading agents are implemented in Python and employ random, SA, and TD3 algorithms. We used the Python random library to generate the offloading agents for the random algorithm. We examine the impact of the system size on the number of agents and the above-mentioned algorithms using various numbers of AN-MEC sites.

5.2. Preliminary comparison of reinforcement learning algorithms

For the preliminary analysis, we compared RL algorithms for continuous search space problems such as DDPG, SAC, and TRPO in the context of V2X with RSU ratio-based offloading optimization. Fig. 6a compares the performance of four RL algorithms—TD3, DDPG, SAC, and TRPO—on an offloading optimization task. TD3 shows superior performance, converging to higher rewards with a better learning rate, due to its use of two critics to mitigate overestimation bias. SAC performs stably with minimal variance, maintaining a consistent reward level throughout. DDPG, while highly unstable early on, improves over time and eventually surpasses SAC, making it suitable for tasks that can tolerate initial instability in favor of higher long-term rewards. TRPO, though slow to improve, reaches a final reward level comparable to SAC and DDPG,

Table 4
Parameter setting.

Parameters	Notations	Values
Topology:		
Number of AN-MEC sites	M	1,2,3,4,5
Number of RSU sites at ith AN-MEC	N_i	9
Number of vehicles at jth RSU of i AN-MEC	$V_{i,j}$	$\{10 \leq 20\}, \{20 \leq 30\}, \{30 \leq 40\}, \{40 \leq 50\}$
Total number of sites	$M(N_i)$	10, 20, 30, 40, 50
Distance between AN-MECs	$d^{AA}_{i \leftrightarrow m}$	$\approx$ 30km – 60km
RAN capacity:		
Uplink, downlink on RSUs site	$B^{VR}_{i,j}, B^{RV}_{i,j}$	10 Gbps
Bandwidth between RSUs	$B^{RR}_{i,j \leftrightarrow n}$	10 Gbps
Uplink, downlink between RSU and AN-MEC	$B^{RA}_{i,j}, B^{AR}_{i,j}$	10 Gbps
Site capacity:		
AN-MEC site	μ^{A}_{i}	$30 \times 10^4 Packets/s$
RSU	$\mu^{R}_{i,j}$	$3 \times 10^4 Packets/s$
Traffic:		
Normal traffic rate	$\lambda_{i,j}$	$\{10 \leq 100\}$
Packet size	p	1Kb (request), 10 Kb (response)



(a) Learning rate (b) Reward

(c) Decision time (d) Convergence time

Fig. 6. Comparison of RL-Based Algorithms for Ratio-Based Offloading Optimization in V2X.

Table 5
SA algorithm settings.

Parameters	Value
Initial temperature (T)	5000
Number of temperatures (R)	300
Neighbors to visit at a temp. (S)	30
Cooling parameter (c)	0.85
Search step (α)	0.1

Table 6
TD3 algorithm settings.

Parameters	Value
Actor and Critic networks parameters	
Number of hidden layer	2
Number of head layer	3
Activation function	Softmax
Input Actor network	Current observation
Output of the Actor	Offloading ratio
Input Critic networks	Batch states and actions
Output of the Critics	Q-value
Training parameters	
Optimizer	ADAM
Batch size	32
Exploration noise	0.1
Discount (γ)	0.99
Tau (τ)	0.005
Policy noise (ϵ)	0.05
Noise clip	0.5
Policy frequency	4
Experiment setups	
Number of episodes	10000
Number of steps each episode	10
Number of experiments	10

indicating it is better suited for tasks prioritizing stability over rapid optimization.

The analysis of RL algorithms' scalability (TD3, TRPO, DDPG, and SAC) in Figs. 6b, 6c, and 6d reveals key insights as the number of MEC sites increases. TD3 consistently achieves the highest rewards up to 50 MEC sites, with a peak reward of 1.66, but experiences a sharp decline to 0.31 at 80 sites and 0.15 at 100 sites. This drop is due to the increasing complexity of the state and action space, as the growing number of sites expands the capacity and latency information fed to the actor and critic networks, while the action space also scales. The actor network, using a Softmax output, must adjust to the higher-dimensional action space where each output element represents the ratio of traffic offloaded to a site, complicating optimization. SAC exhibits a similar pattern, peaking at 1.62 at 50 sites but dropping to 0.19 at 100 sites, indicating that both algorithms struggle with scalability as the number of MEC sites increases. In contrast, DDPG peaks unexpectedly at 1.52 at 30 sites, only to plummet to 0.01 by 80 sites, due to its single-critic structure, which makes it prone to overfitting and less capable of managing the increasing complexity of the state-action space. TRPO, though slower to improve, reaches comparable rewards to SAC and DDPG at higher site counts (approximately 0.42 at 80 and 100 sites), suggesting that while its conservative policy updates may slow progress, it may better accommodate the complexities of larger environments.

When evaluating decision time (Fig. 6c), TRPO exhibits inefficiency with a significant spike to 0.037 ms at 80 sites, indicating that its decision-making process struggles with scalability due to the complexity of updating its policy structure as the action space increases. In contrast, TD3, SAC, and DDPG show more consistent and predictable increases in decision time, with SAC having the lowest values at higher site counts (e.g., 0.015 ms at 100 sites). However, the differences between TD3, SAC, and DDPG remain minor, suggesting that all three algorithms

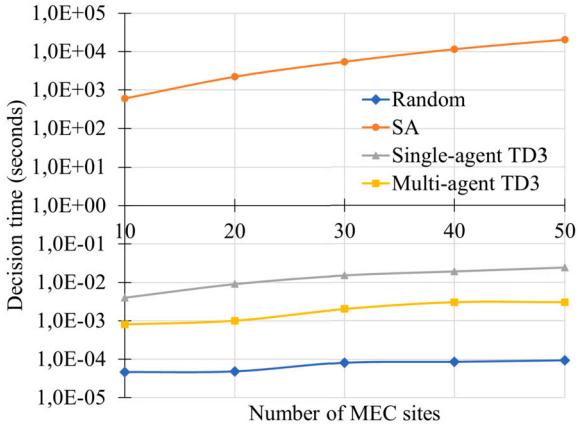


Fig. 7. Decision time.

maintain reasonable efficiency in real-time decision-making even as the number of MEC sites increases.

Finally, in terms of convergence time (Fig. 6d), TD3 exhibits relatively low convergence times, even with increasing site counts (e.g., 2325 s at 100 sites), making it more practical in terms of training efficiency. SAC, however, shows dramatically longer convergence times, especially beyond 50 sites (e.g., 34159 s at 100 sites), indicating that its training complexity grows significantly with scalability. DDPG and TRPO also show longer convergence times at higher site counts, though TRPO manages better than DDPG in environments with a larger number of sites. This indicates that while SAC and TD3 have strengths in decision time, their convergence time makes TD3 a better choice overall in large-scale scenarios where faster training is critical.

5.3. Evaluation of SA and TD3 for ratio-based offloading in V2X systems with RSU

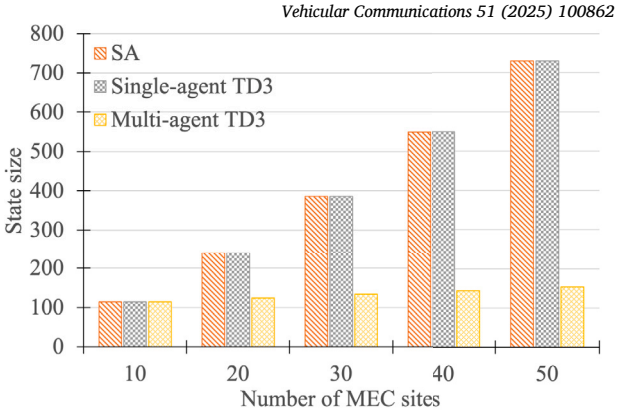
The experimental results are organized to address the following key aspects: decision time, convergence time, decision quality, and offloading with unknown parameters, comparing RL and SA in each case, as well as the comparison between single-agent and multi-agent offloading.

5.3.1. Decision time

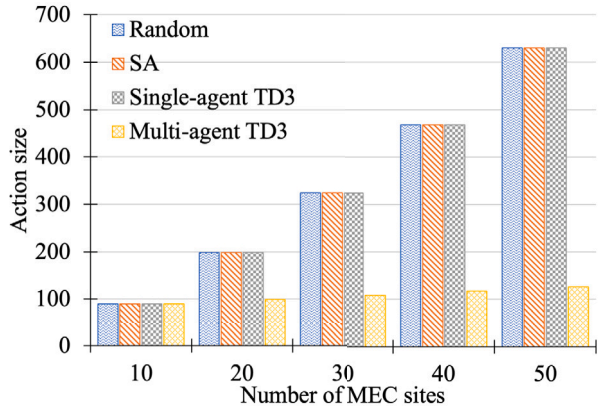
Decision time refers to the duration an agent needs to determine an offloading action. As a control plane component, the offloading agent must forward traffic or tasks within seconds to accommodate fluctuating traffic arrival rates. This study compares the decision times of TD3, SA, and random-based algorithms in determining the offloading ratio.

As illustrated in Fig. 7, SA required 602 to 20,421 seconds to calculate the offloading ratio for a V2X system with 10 to 50 sites using exhaustive searching. This decision time far exceeds the control plane's requirements. In contrast, single-agent and multi-agent TD3 could map the input to an offloading ratio within 4 to 24 milliseconds and 0.8 to 3 milliseconds, respectively, for V2X systems with 10 to 50 sites. This meets control plane requirements and is five orders of magnitude faster than SA. TD3-based agents can simultaneously decide and train the offloading model, using neural networks to map the information of the system to the offloading ratio. A random agent, employing a random-based algorithm, decided the offloading ratio within 46 to 95 microseconds. However, due to its random nature, it can assign traffic to a site with insufficient capacity.

We compared the three algorithms on various scales of the system, affecting the input and output sizes, as shown in Fig. 8 a and b. In Fig. 8a, the state represents the information of the V2X system, including the communication and computation capacity, which is used by the agent as input to decide the best action in the form of offloading ratios. Multi-



(a) State space.



(b) Action space.

Fig. 8. State and action space comparison.

agent systems consider local environment states within an AN-MEC site to decide an action, thereby showing the least input size.

Fig. 8b shows the agent action size, which represents the offloading ratio. This ratio describes the proportion of traffic offloaded to RSU and AN-MEC sites. In a single-agent setup, placed in the CO, the agent decides the offloading ratio for all RSUs. Conversely, in a multi-agent setup, each agent is collocated with an AN-MEC and decides the offloading ratio for all RSUs within an AN-MEC site. This configuration results in smaller output sizes for multi-agents. The smaller input and output sizes of the multi-agent system also contribute to significantly reduced decision times, approximately one order of magnitude lower than the single-agent system.

5.3.2. Convergence time

Convergence time is the amount of time required for an algorithm to produce optimal results with the lowest average latency. Fig. 9 depicts a comparison of the convergence time between SA and TD3. In the beginning, the TD3-agent utilized weightedan ANng because depending on a random float to determine the initial offloading ratio would require a momentous time to converge.

As illustrated in Fig. 9, TD3's convergence time is one order of magnitude faster than that of SA due to the model's pre-training using weighted-based offloading. Single-agent and multi-agent systems TD3 required 283 and 89 seconds, respectively, to converge in the V2X system with 10 sites and 764 and 178 seconds with 50 sites. In contrast, SA required 2,205 and 20,421 seconds to converge in systems with 10 and 50 sites, respectively.

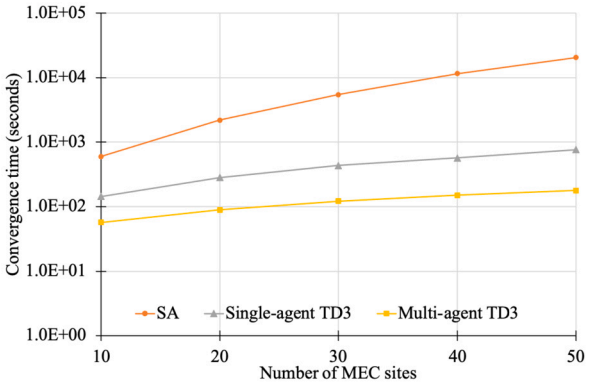


Fig. 9. Convergence time.

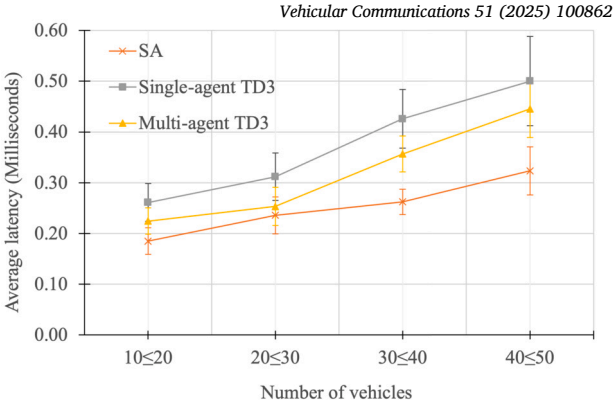


Fig. 10. Offloading performance.

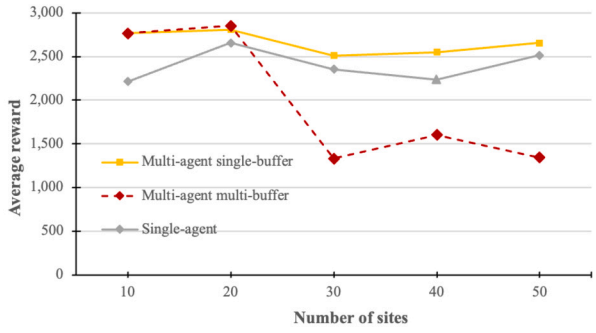


Fig. 11. Single vs. multi-buffer performance.

The SA runs for each AN-MEC (M), and for each base station, the algorithm visits each RSU (N). Inside, it runs for R iterations (temperature decreases), and for each iteration, it performs S neighbor searches. Each search involves capacity evaluation and residual capacity calculations, which are linear in the number of RSUs and AN-MECs. Thus, the overall time complexity is $O(M \times N \times R \times S \times (M + N))$, simplified to $O(M \times N^2 \times R \times S)$ when considering the dominant terms. This complexity reflects the combinatorial nature of the offloading decision problem and the iterative nature of the simulated annealing process. The SA's convergence time increased significantly as the number of sites increased. TD3-based offloading uses NN, whose complexity depends on the number of inputs n, epochs t, and the number of hidden layers k. Within the NN layer, there is a matrix multiplication with a complexity of m^3, where m is the size of the matrices that are multiplied. The complexity of NN training is $O(nt(km^3))$ or $O(nt(m^3))$ if the constant k is removed. The NN operation has a linear relationship with the input n. As depicted in Fig. 9, the decision time of the TD3 offloading agent increased slightly as the number of sites increased. In this study, a weighted-based offloading algorithm is used instead of the random action typically used by the RL algorithm in the initial training iterations to collect experience data.

5.3.3. Decision quality

The performance of the offloading agent is measured by the average latency, defined as the time it takes for all traffic sent by vehicles to receive a response. To measure the agent's performance, we defined four arrival traffic scenarios. Each scenario consists of a different number of vehicles that generate traffic to the RSU. We do not impose decisions on vehicles since they have limited resources, and determining the offloading ratio involves energy-intensive environmental monitoring. In this instance, the environment data are fairly broad and include fog (RSU) and edge information (AN-MEC).

Fig. 10 compares the performance of SA and TD3-based offloading agents. The SA that performs extensive searching has the highest performance, requiring 0.18 and 0.32 seconds to serve 10-20 and 40-50 vehicles, respectively. Single-agent TD3 had an average latency of 0.26 ms and 0.5 ms for serving 10-20 and 40-50 vehicles, respectively. In comparison, multi-agent TD3 had a lower average latency of 0.22 ms and 0.45 ms for the same vehicle ranges.

The average latency of the single-agent and multi-agent TD3 is only 0.2 and 0.1, greater than that of the SA. Due to their low propagation delay, nearby available resources are more likely to be offloaded than distant ones. The multi-agent TD3 that utilizes local data is more efficient than the single-agent TD3 that utilizes global data. An offloading agent may not require specific information, such as information on a remote RSU site. This useless feature is eliminated, making the training process more efficient and producing better outcomes than single-agent TD3.

To train the TD3 models, the multi-agent utilized a centralized replay buffer. Fig. 11 compares the performance of the multi-agent using a centralized replay buffer with those using single-replay and multi-replay buffers. Using multi-replay buffers in the multi-agent system introduces unknown information from traffic offloaded from other agent coverage, such as traffic from other AN-MEC sites. As the number of sites increases, the unknown information also increases, resulting in degraded performance. Fig. 11 illustrates that using multi-replay buffers causes the performance of the multi-agent to degrade by almost half compared to using a single buffer.

Fig. 12 shows the traffic distributions for all offloading agents. When the number of vehicles increased, more traffic is directed to AN-MEC sites by SA and multi-agent TD3. In terms of computing latency, a centralized large capacity such as AN-MEC outperforms a distributed with small capacity such as RSU [53]. SA, with the best performance, directed 52% traffic to AN-MEC sites, and 48% is served in RSUs in serving 40-50 vehicles. With a ratio of 66%:34%, single-agent TD3 sent a large amount of traffic to RSUs. In contrast, multi-agent TD3 sent a high proportion of traffic to AN-MEC sites (36%:64%). In this optimization, directing traffic to upper tiers will increase propagation latency, while directing a high volume of traffic to lower tiers increases computation latency due to the lower capacity of the bottom tier. Finding the correct balance of referred traffic leads to optimal performance.

The RSU has a capacity that is ten times smaller than that of the AN-MEC. As a result, when vehicle traffic is directed to the RSU, it quickly becomes occupied, resulting in a low horizontal offloading ratio (H-R), as shown in Fig. 13. The horizontal offloading between RSUs is less than 10%. For single-agent TD3, the percentage of traffic served at the arrival RSU (V-R) is 45%, while it is 47% for multi-agent TD3 and 29% for SA. However, AN-MEC sites are preferred due to their higher capacity, proximity to users, and centralization, and therefore, the total traffic served at AN-MEC sites managed by SA and multi-agent TD3 is higher than at RSU sites. This traffic includes both vertical (V-AN) and horizontal (H-AN) offloading to AN-MEC sites.

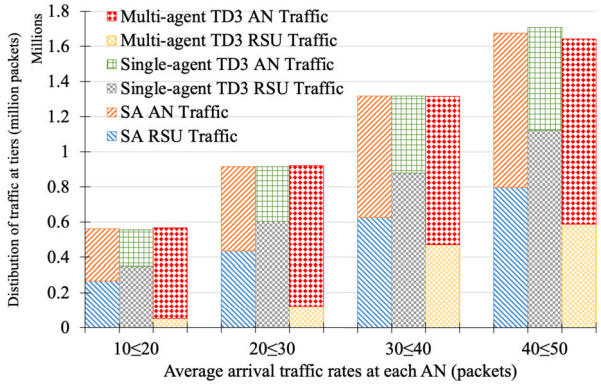


Fig. 12. Traffic distribution.

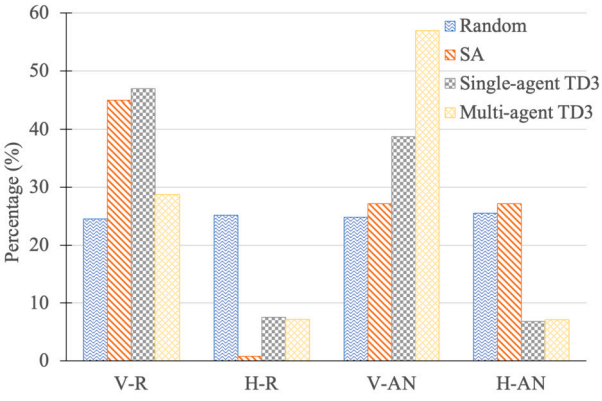
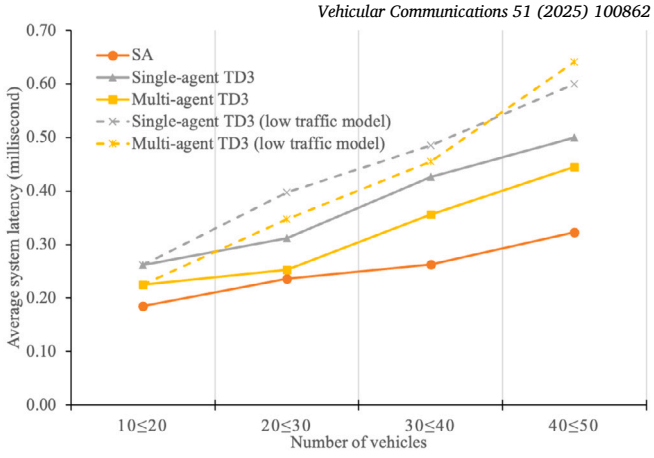
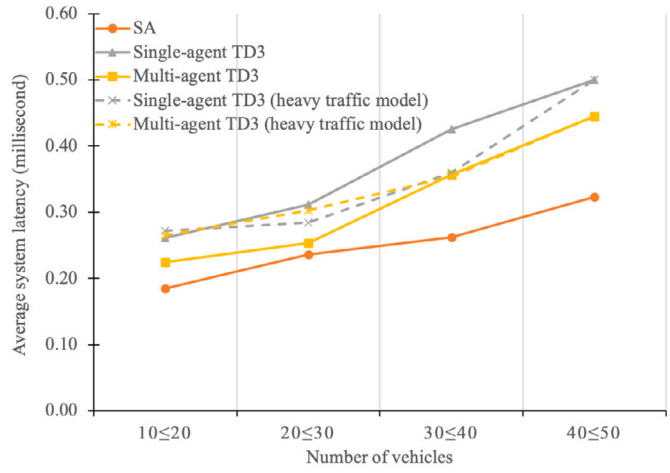


Fig. 13. Offloading ratio.



(a) Low traffic model performance.



(b) Heavy traffic model performance.

Fig. 14. Effect of fluctuating traffic to the agent model.

5.3.4. Decision with unknown information

The agent may encounter unknown information, such as fluctuating heavy traffic when trained on light traffic. When the environment state changes, conventional optimization recalculates the offloading decision. Fig. 14 shows the responsiveness to traffic fluctuations. In Fig. 14a, the TD3 model trained on light traffic handles heavy traffic fluctuations, whereas in Fig. 14b, the TD3 model trained on heavy traffic manages incoming light traffic.

The model trained to handle heavy traffic is more transferable to light traffic than the model trained to handle light traffic to heavy traffic. 14a demonstrates the failure of the light traffic model of single-agent and multi-agent TD3 to handle heavy arrival traffic. The latency gap between the correct model and the light model is up to 0.28 milliseconds.

In contrast, the decision made by the model trained by heavy traffic had better performance than the original one, shown in 14b of single-agent TD3. Serving heavy arrival traffic from 30-40 vehicles and 40-50 vehicles, the performance of the multi-agent TD3 model trained with the heavy model is essentially equal to that of the model trained with the specified arrival rate. However, when the model is applied to low arrival rates with 10-20 and 20-30 vehicles, it performs worse than the correct model. This may be due to the different traffic patterns caused by the random number of vehicles that generate traffic to specific locations.

6. Conclusion

In the V2X with the RSU system, an AN-MEC placed at the base station could extend the RSU to create a two-tier fog-edge architecture. Offloading is important for distributing arriving traffic within the system and reducing the average delay. Offloading was addressed as a control plane problem with a short decision time to accommodate hotspot traffic. TD3 with NN to map the input to offloading ratio has a decision time of unit milliseconds, which is five orders of magnitude faster than SA. The latter takes 602 to 20,421 seconds to determine the offloading ratio, which may be too slow for future traffic. Single-agent TD3 decides the offloading ratio in 4 to 24 milliseconds for systems with 10 and 50 sites, respectively. Multi-agent TD3, with smaller input and output, has even faster decision times of 1 and 3 milliseconds. In all traffic scenarios, the average latency experienced by the arrival traffic using SA was 0.18 to 0.32 milliseconds. By initial training using weighted-based offloading, TD3 can approximate SA's performance. The average latency for single-agent TD3 was 0.26 to 0.5 milliseconds, while that of multi-agent TD3 was 0.22 to 0.45 milliseconds. In TD3-based agents, the model trained by heavy traffic is more adaptable to other traffic patterns. Most of the traffic was forwaran ANites, which are more centralized, have greater capacity than RSU, and are also closer to users (vehicles). Due to the small capacity of the RSU, the horizontal offloading ratio between them is also modest.

For future work, addressing the longer convergence times in early iterations of RL by incorporating imitation learning could streamline the process. Furthermore, validating these models through real testbed experiments will be crucial to assessing their practical efficacy in V2X scenarios. Exploring the impact of power consumption on offloading strategies and integrating it as a key metric will enhance the sustainability and efficiency of V2X systems.

CRediT authorship contribution statement

Widhi Yahya: Writing – original draft, Visualization, Software, Methodology, Investigation, Formal analysis. **Ying-Dar Lin:** Validation,

Supervision, Methodology, Conceptualization. **Faysal Marzuk:** Writing – review & editing, Validation, Methodology. **Piotr Chołda:** Validation, Supervision, Formal analysis. **Yuan-Cheng Lai:** Supervision, Methodology, Investigation, Formal analysis.

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.

Acknowledgement

This work was supported in part by the National Science and Technology Council, Taiwan, under grants 111-2221-E-A49-095-MY3, and in part by the PL-Grid infrastructure: http://www.plgrid.pl/en.

Data availability

No data was used for the research described in the article.

References

[1] M.H. Garcia, A. Molina-Galan, M. Boban, J. Gozalvez, B. Coll-Perales, T. Sahin, A. Kousaridas, A tutorial on 5G NR V2X communications, IEEE Commun. Surv. Tutor. 23 (2021) 1972–2026.

[2] M. Harounabadi, D.M. Soleymani, S. Bhadauria, M. Leyh, E. Roth-Mandutz, V2X in 3GPP standardization: NR sidelink in release-16 and beyond, IEEE Commun. Stand. Mag. 5 (1) (2021) 12–21.

[3] ETSI, 5G; Vehicle-to-everything (V2X); Media handling and interaction (3GPP TR 26.985 version 16.0.0 Release 16), Technical Report ETSI TR 126 985 V16.0.0, European Telecommunications Standards Institute, April 2020.

[4] P.L. Nguyen, R.H. Hwang, P.M. Khiem, K. Nguyen, Y.D. Lin, Modeling and minimizing latency in three-tier V2X networks, in: Proceeding of IEEE Global Communications Conference, GLOBECOM 2020 2020-January, 12 2020.

[5] S. González, A. de la Oliva, X. Costa-Pérez, A. Di Giglio, F. Cavaliere, T. Deiß, X. Li, A. Mourad, 5G-Crosshaul: an SDN/NFV control and data plane architecture for the 5G integrated Fronthaul/Backhaul, Trans. Emerg. Telecommun. Technol. 27 (9) (2016) 1196–1205.

[6] 3rd Generation Partnership Project (3GPP), 5G; Service requirements for next generation new services and markets, Technical Specification 3GPP TS 22.261, 3GPP, release 15, 2018.

[7] A. Shakarami, M. Ghobaei-Arani, A. Shahidinejad, A survey on the computation offloading approaches in mobile edge computing: a machine learning-based perspective, Comput. Netw. 182 (2020) 107496.

[8] B. Kar, W. Yahya, Y.-D. Lin, A. Ali, Offloading using traditional optimization and machine learning in federated cloud–edge–fog systems: a survey, IEEE Commun. Surv. Tutor. 25 (2) (2023) 1199–1226.

[9] S. Fujimoto, H.V. Hoof, D. Meger, Addressing function approximation error in actor-critic methods, in: 35th International Conference on Machine Learning, ICML 2018 4, 2018, pp. 2587–2601.

[10] C. Sonmez, C. Tunca, A. Ozgovde, C. Ersoy, Machine learning-based workload orchestrator for vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (2021) 2239–2251.

[11] S. Zhou, Y. Sun, Z. Jiang, Z. Niu, Exploiting moving intelligence: delay-optimized computation offloading in vehicular fog networks, IEEE Commun. Mag. 57 (2019) 49–55.

[12] H. Ke, J. Wang, L. Deng, Y. Ge, H. Wang, Deep reinforcement learning-based adaptive computation offloading for MEC in heterogeneous vehicular networks, IEEE Trans. Veh. Technol. 69 (2020) 7916–7929.

[13] J. Shi, J. Du, J. Wang, J. Yuan, Deep reinforcement learning-based v2v partial computation offloading in vehicular fog computing, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), vol. 2021-March, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 1–6.

[14] D. Yang, B. Ni, H. Qin, F. Ma, Adaptive Task Offloading in V2X Networks Based on Deep Reinforcement Learning, SPIE-Intl Soc Optical Eng, 2022, pp. 579–590.

[15] W. Chen, X. Wei, K. Chi, K. Yu, A. Tolba, S. Mumtaz, M. Guizani, Computation time minimized offloading in noma-enabled wireless powered mobile edge computing, IEEE Trans. Commun. (2024) 7182–7197.

[16] N. Wang, S. Pang, X. Ji, M. Wang, S. Qiao, S. Yu, Intelligent driving task scheduling service in vehicle-edge collaborative networks based on deep reinforcement learning, IEEE Trans. Netw. Serv. Manag. (2024) 4357–4368.

[17] M. Khayyat, I.A. Elgendy, A. Muthanna, A.S. Alshahrani, S. Alharbi, A. Koucheryavy, Advanced deep learning-based computational offloading for multilevel vehicular edge-cloud computing networks, IEEE Access 8 (2020) 137052–137062.

[18] H. Zhang, Z. Wang, K. Liu, V2X offloading and resource allocation in SDN assisted MEC-based vehicular networks, China Commun. 17 (2020) 266–283.

[19] B. Fan, Z. He, Y. Wu, J. He, Y. Chen, L. Jiang, Deep learning empowered traffic offloading in intelligent software defined cellular V2X networks, IEEE Trans. Veh. Technol. 69 (2020) 13328–13340.

[20] A. Paul, K. Singh, C.-P. Li, O.A. Dobre, T.Q. Duong, Digital twin-aided vehicular edge network: a large-scale model optimization by quantum-drl, IEEE Trans. Veh. Technol. (2024) 1–17.

[21] B. Qiu, Y. Wang, H. Xiao, Z. Zhang, Deep reinforcement learning-based adaptive computation offloading and power allocation in vehicular edge computing networks, IEEE Trans. Intell. Transp. Syst. (2024) 1–11.

[22] G. Wu, Z. Liu, M. Fan, K. Wu, Joint task offloading and resource allocation in multi-uav multi-server systems: an attention-based deep reinforcement learning approach, IEEE Trans. Veh. Technol. 73 (8) (2024) 11964–11978.

[23] B. Hu, W. Zhang, Y. Gao, J. Du, X. Chu, Multi-agent deep deterministic policy gradient-based computation offloading and resource allocation for Isac-aided 6g v2x networks, IEEE Int. Things J. (2024) 33890–33902.

[24] Y. Cong, M. Liu, C. Wang, S. Sun, F. Hu, Z. Liu, C. Wang, Task scheduling and power allocation in multiuser multiserver vehicular networks by noma and deep reinforcement learning, IEEE Int. Things J. 11 (13) (2024) 23532–23543.

[25] Y. Li, L. Li, Computation offloading and resource allocation in mec-enabled vehicular networks: partial offloading versus binary offloading, in: 2024 7th World Conference on Computing and Communication Technologies (WCCCT), 2024, pp. 260–266.

[26] Y. Guo, D. Ma, H. She, G. Gui, C. Yuen, H. Sari, F. Adachi, Deep deterministic policy gradient-based intelligent task offloading for vehicular computing with priority experience playback, IEEE Trans. Veh. Technol. 73 (7) (2024) 10655–10667.

[27] B. Wang, D. Tu, J. Wang, Drl-based partial task offloading for multiple vehicles in vec networks, in: 2024 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB), 2024, pp. 1–6.

[28] M. Yan, R. Xiong, Y. Wang, C. Li, Edge computing task offloading optimization for a uav-assisted Internet of vehicles via deep reinforcement learning, IEEE Trans. Veh. Technol. 73 (4) (2024) 5647–5658.

[29] P. Qin, Y. Fu, R. Ding, H. He, Competition-awareness partial task offloading and uav deployment for multitier parallel computational Internet of vehicles, IEEE Syst. J. (2024) 1–12.

[30] B. Li, L. Zhu, L. Tan, A distributed deep reinforcement learning-based optimization scheme for vehicle edge computing task offloading, in: 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2024, pp. 218–223.

[31] Y. Chen, X. Li, G. Xu, L. Liu, X. Wang, V.C.M. Leung, Competitive and cooperative computation offloading for intensive heterogeneous tasks in vehicular edge computing networks, in: ICC 2024 - IEEE International Conference on Communications, 2024, pp. 5491–5496.

[32] W. Fan, Y. Zhang, G. Zhou, Y. Liu, Deep reinforcement learning-based task offloading for vehicular edge computing with flexible rsu-rsu cooperation, IEEE Trans. Intell. Transp. Syst. 25 (7) (2024) 7712–7725.

[33] Y. Hou, Z. Wei, R. Zhang, X. Cheng, L. Yang, Hierarchical task offloading for vehicular fog computing based on multi-agent deep reinforcement learning, IEEE Trans. Wirel. Commun. 23 (4) (2024) 3074–3085.

[34] X. Ning, M. Zeng, Z. Fei, Joint task offloading and service migration in ris assisted vehicular edge computing network based on deep reinforcement learning, in: 2024 International Conference on Computing, Networking and Communications (ICNC), 2024, pp. 1037–1042.

[35] P. Qin, Y. Wang, Z. Cai, J. Liu, J. Li, X. Zhao, Madrl-based urllc-aware task offloading for air-ground vehicular cooperative computing network, IEEE Trans. Intell. Transp. Syst. 25 (7) (2024) 6716–6729.

[36] S.S. Shinde, D. Tarchi, Hierarchical reinforcement learning for multi-layer multi-service non-terrestrial vehicular edge computing, IEEE Trans. Mach. Learn. Commun. Netw. 2 (2024) 1045–1061.

[37] N. Fofana, A.B. Letaifa, A. Rachedi, Intelligent task offloading in vehicular networks: a deep reinforcement learning perspective, IEEE Trans. Veh. Technol. (2024) 1–16.

[38] S. Li, W. Sun, Q. Ni, Y. Sun, Road side unit-assisted learning-based partial task offloading for vehicular edge computing system, IEEE Trans. Veh. Technol. 73 (4) (2024) 5546–5555.

[39] M. Li, J. Gao, L. Zhao, X. Shen, Deep reinforcement learning for collaborative edge computing in vehicular networks, IEEE Trans. Cogn. Commun. Netw. 6 (2020) 1122–1135.

[40] L. Zhu, Z. Zhang, L. Liu, L. Feng, P. Lin, Y. Zhang, Online distributed learning-based load-aware heterogeneous vehicular edge computing, IEEE Sens. J. 23 (15) (2023) 17350–17365.

[41] L. Cui, C. Guo, C. Wang, Collaborative edge computing for vehicular applications modeled by general task graphs, in: 2023 4th Information Communication Technologies Conference (ICTC), 2023, pp. 265–270.

[42] Z. Wei, B. Li, R. Zhang, X. Cheng, L. Yang, Many-to-many task offloading in vehicular fog computing: a multi-agent deep reinforcement learning approach, IEEE Trans. Mob. Comput. 23 (3) (2024) 2107–2122.

[43] P. Dai, Y. Huang, K. Hu, X. Wu, H. Xing, Z. Yu, Meta reinforcement learning for multi-task offloading in vehicular edge computing, IEEE Trans. Mob. Comput. 23 (3) (2024) 2123–2138.

[44] B. Hazarika, K. Singh, S. Biswas, S. Mumtaz, C.-P. Li, Multi-agent DRL-based task offloading in Mmultiple RIS-aided IoV networks, IEEE Trans. Veh. Technol. 73 (1) (2024) 1175–1190.

[45] N.H. Nguyen, P.L. Nguyen, H. Dinh, T.H. Nguyen, K. Nguyen, Multi-Agent Multi-Armed Bandit Learning for Offloading Delay Minimization in V2X Networks, IEEE, 2021, pp. 47–55.

[46] F.G. Wakgra, B. Kar, S.B. Tadele, S.-H. Shen, A.U. Khan, Multi-objective offloading optimization in mec and vehicular-fog systems: a distributed-td3 approach, IEEE Trans. Intell. Transp. Syst. (2024) 1–13.

[47] S. Chen, J. Hu, Y. Shi, Y. Peng, J. Fang, R. Zhao, L. Zhao, Vehicle-to-everything (V2X) services supported by LTE-based systems and 5G, IEEE Commun. Stand. Mag. 1 (2017) 70–76.

[48] C. Consulting, GSMA, Vision 2030: low-band spectrum for 5g, maximising the socioeconomic value of spectrum, https://www.gsma.com/spectrum/wp-content/uploads/2022/07/Low-Band-Spectrum-for-5G.pdf, 2022.

[49] L. Huawei Technologies Co., 5g network architecture: a high-level perspective, Tech. Rep., Huawei, 2016, https://www-file.huawei.com/~/media/CORPORATE/PDF/mbb/5g_nework_architecture_whitepaper_en.pdf?la=en. (Accessed 13 July 2024).

[50] Y. Li, Optimization of task offloading problem based on Simulated-Annealing algorithm in MEC, in: 2021 9th International Conference on Intelligent Computing and Wireless Optical Communications (ICWOC), 2021, pp. 47–52.

[51] L.M. Rere, M.I. Fanany, A.M. Arymurthy, Simulated annealing algorithm for deep learning, Proc. Comput. Sci. 72 (2015) 137–144.

[52] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: a next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 2623–2631.

[53] W. Yahya, E. Oki, Y.-D. Lin, Y.-C. Lai, Scaling and offloading optimization in pre-CORD and post-CORD multi-access edge computing, IEEE Trans. Netw. Serv. Manag. 18 (4) (2021) 4503–4516.