A Blockchain-Enabled Vehicular Edge Computing Framework for Secure Performance-oriented V2X Service Delivery

Mohammad Fardad, *Student Member, IEEE,* Gabriel-Miro Muntean, *Fellow, IEEE,* and Irina Tal, *Member, IEEE*

***Abstract*—Vehicular Edge Computing (VEC) has emerged as a promising paradigm to enable low-latency Vehicle-to-Everything (V2X) services by bringing computing resources closer to vehicles. However, the high dynamicity of vehicular networks poses significant challenges in designing an optimal policy for delivering V2X services while ensuring security and timely service delivery. To address these challenges, this paper proposes a Blockchain-Enabled Vehicular Edge Computing (BEVEC) framework that employs a dual-layer verification process empowered with a permissioned blockchain to ensure data accuracy and integrity. A novel system utility function is designed to measure the performance of the BEVEC, which also serves as the basis for a consensus mechanism of the permissioned blockchain. To optimize this utility, a Deep Reinforcement Learning (DRL) algorithm is proposed to enable timely service delivery in BEVEC. Simulation-based results demonstrate the effectiveness of the proposed algorithm when compared to existing approaches. On average, it obtained an 18% reduction in latency, a 38% improvement in successful service delivery, and a 65% decrease in energy consumption.**

***Index Terms*—Vehicular edge computing, blockchain, vehicle-to-everything, deep-reinforcement learning, latency.**

I. Introduction

VEHICLE-TO-EVERYTHING (V2X) communication is a key paradigm which enables seamless communication between neighboring road users (including vehicles, pedestrians, and infrastructure) and fosters advancements in road traffic efficiency and provision of innovative services such as autonomous driving, and onboard rich media entertainment [1]–[3]. V2X applications exhibit unique characteristics such as varied payload sizes and are associated with different types of traffic with different priorities, as well as various quality of service (QoS) requirements, including maximum tolerable latency, reliability, and data rate. The primary concern of V2X applications, particularly those related to safety, is related to the latency, which decreases safety levels as the delays in receiving safety information increase. However, certain V2X applications, such as specific vehicular tasks like computation offloading, may place a greater emphasis on other QoS requirements than on latency in applications such as sharing high-definition maps, augmented reality and virtual reality, online gaming, etc. [4], [5]. A possible approach to increasing QoS is to integrate more advanced computing and storage resources into vehicles [6]. However, the limited physical space and the high costs associated with providing these additional resources make it challenging to ensure efficient and stable execution of any associated onboard applications [7]. The latter cannot be accommodated without increasing manufacturing costs, which is not desirable [8].

Traditionally, cloud computing has been utilized to handle computationally intensive tasks in mobile networks. However, cloud computing yields high response times, which are unsuitable for dynamic and latency-critical environments such as vehicular networks [9]. To address these issues, multi-access edge computing (MEC) has emerged as a promising solution, particularly in the context of vehicular networks, where it is referred to as Vehicular Edge Computing (VEC) [9], [10]. VEC effectively deploys several computing and storage resources in close proximity of vehicles, making use of base stations (BS) and roadside units (RSU) to deliver robust V2X services. Although VEC servers deployed on RSUs and BSs can reduce connectivity latency due to their proximity to vehicles, their limited computational and communication resources require optimized resource management strategies. Furthermore, the limited coverage range of RSUs restricts the number of vehicles that can access their V2X services without incurring additional delays, mostly due to the VEC handover and migration of services [9], [11]. To address these limitations, vehicles with sufficient resources can also be employed as VEC servers to support V2X applications [4], [7], [12]. However, the mobility of vehicles increases the complexity of service delivery in such a context. Furthermore, sensitive and private vehicle information, as well as data migration between different VEC servers, present potential security breaches and data leak vectors [13]. As a result, V2X service delivery is a multifaceted and intricate issue that must be carefully managed.

Blockchain has been integrated with VEC to ensure the security and privacy of V2X applications [14]–[16]. By incorporating blockchain technology into VEC to support V2X services, it is possible to create a highly effective and secure data-sharing infrastructure among VEC servers. This system not only facilitates the provision of information about adjacent service providers to vehicles, but it also improves collaboration and security within the network, enhancing the delivery of V2X services. However, the consensus mechanism employed

The authors are with the Lero SFI Centre for Software, Dublin City University, Ireland (e-mail: mohammad.fardad2@mail.dcu.ie; gabriel.muntean@dcu.ie; irina.tal@dcu.ie).

This work was supported by the Science Foundation Ireland grant 13/RC/2094_P2 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Science Foundation Ireland Research Centre for Software (www.lero.ie).

Manuscript received ; revised .

by blockchain introduces additional energy consumption and delays in the vehicular network [17]. In a public blockchain, every distributed node is obliged to participate in the consensus procedure, resulting in longer duration of both block generation and verification, and generating higher energy consumption. These increased delay and energy consumption are not suitable for the energy-constrained and delay-sensitive vehicular networks. As a result, some studies [18], [19] have opted for a permissioned blockchain approach within the VEC system. Through this approach, only a distinct subset of nodes is given permission to participate in the blockchain consensus process, resulting in a notably faster overall procedure.

While researchers started to investigate the potential of blockchain technology in enhancing data security within VEC systems, to the best of our knowledge, its potential was not yet explored within a versatile, all-encompassing V2X service delivery platform. Furthermore, none of the existing solutions take into account the critical aspect of traffic prioritization of V2X applications for obtaining high QoS levels in the context of low latency and reliable service delivery.

In this context, this paper makes the following contributions:

- Proposes a novel framework called **BEVEC**: **B**lockchain-**E**nabled **V**ehicular **E**dge **C**omputing for secure, performance-oriented V2X service delivery. BEVEC prioritizes V2X application traffic and delivers services in a specified time, ensuring that critical applications receive the resources they need and are executed on time.
- Introduces a dual-layer verification process inside BEVEC to ensure secure and reliable delivery of general-purpose V2X services. The first layer, local verification, guarantees the accuracy of exchanged data, while the permissioned blockchain of the second layer checks for data integrity. This two-tier approach provides comprehensive security for V2X services, ensuring the authenticity and reliability of information exchanged in the vehicular environment.
- Proposes a novel system utility function that takes into account three key factors: consumed energy, exchanged data size, and priority of V2X applications traffic. This function serves as a measure of system performance and also as a basis for selecting block verifier nodes in the consensus mechanism. The goal is to achieve reliable and low-latency delivery of V2X services.
- Describes a novel Deep Reinforcement Learning (DRL) algorithm, named 3DPER, to improve the performance of BEVEC in terms of energy consumption, latency, and service delivery success rate. Simulations show how 3DPER outperforms existing methods in terms of these metrics, achieving an average of 18% latency reduction, 38% improvement in successful service delivery, and 65% decrease in energy consumption.

The structure of this paper is as follows. In Section II, we dive into related work. Section III introduces the system model and its underlying assumptions. The proposed solution is elaborated upon in Section IV, and Section V provides an in-depth analysis of performance. Section VI wraps up the paper with our key conclusions. To ease readability, Table I compiles a list of the major notations used in this paper.

TABLE I
PRIMARY NOTIONS

Parameter	Value
$\mathcal{V}/\mathcal{N}$	Set of vehicles/RSUs and BSs also blockchain nodes
$\hat{\mathcal{N}}$	Subset of blockchain nodes
$\mathcal{M}_{ij}$	Message from entity i to j
$\mathcal{S}(\mathcal{M}_{ij})$	Size of message $\mathcal{M}_{ij}$
$c_i/T_i^{max}/\rho_i$	content/maximum tolerable latency/ traffic priority of $\mathcal{M}_{ij}$
τ_{ij}	Transmission latency from entity i to j
τ_j^{proc}	Processing latency of entity j
τ_j^{bv}	blockchain latency
R_{ij}	Communication rate between entity i and j
p_{ij}	Transmission power from entity i to j
h_{ij}	channel coefficient between entity i and j
d_{ij}	channel coefficient between entity i and j

II. RELATED WORK

In this section, we explore the current research landscape concerning the provision of V2X services within a VEC-based environment, categorizing the literature into several key areas. A significant portion of the available research focuses on different aspects of service delivery, including task scheduling, computation offloading, handover between VEC servers, security considerations, blockchain integration, and other unaddressed challenges.

A. Task Scheduling and Computation Offloading in VEC Environments

Researchers have explored extensively optimization algorithms, game-theoretic frameworks, and distributed decision-making approaches to manage efficiently task distribution and resource allocation in VEC networks. For example, Gao *et al.* [20] proposed a two-layer optimization algorithm for joint task offloading and resource allocation in VEC networks, considering QoS constraints. They combine DRL and convex optimization methods to optimize energy and delay reduction. Zhao *et al.* [21] proposed a game-theoretic collaborative computation offloading architecture that integrates edge and cloud computing. The architecture jointly optimizes computation offloading and resource allocation to maximize system utility while minimizing task processing delay. However, the authors restricted their VEC environment model to a single VEC scenario, which does not accurately reflect the diverse range of operational conditions and complexities present in real-world VEC deployments.

Some studies have expanded their focus to include scenarios that involve multiple VECs. For example, Luo *et al.* [22] developed a model for a multi-vehicle, multi-VEC computation offloading framework. They proposed a self-learning distributed computation offloading approach that formulates the computation offloading problem as a distributed decision-making game, where each vehicle is a player that seeks to minimize its overall cost, which includes latency and offloading expenses. Shang *et al.* [23] proposed a combined deep learning and convex optimization method to reduce energy consumption during edge offloading in a scenario with

multiple vehicles and roadside edge servers. Ning *et al*. [24] proposed a VEC framework that optimizes partial computation offloading and uses an adaptive task scheduling algorithm to maximize system-wide profit. They consider the selfish behavior of vehicles and utilize game theory to demonstrate the existence and optimality of the Nash equilibrium. Li *et al*. [25] proposed a DRL-based method to optimize task completion time and energy consumption in VEC considering the priority of tasks.

Despite the introduction of multi-VEC server scenarios and consideration of vehicle mobility in these studies, the researchers overlooked the crucial aspect of handovers between VEC servers to complete in-progress applications and the possible necessity for retransmissions or addressing failures.

In light of this challenge, researchers have shown significant interest in addressing handover between VEC servers. Specifically, [26]–[28] have concentrated on the task offloading and migration of individual vehicles in straightforward scenarios, with their primary emphasis placed on service handover. Unfortunately, these studies did not adequately address the importance of resource competition among multiple vehicles in real-world VEC environments. The challenges of handover in multi-vehicle scenarios have only been explored in a few other research papers [29]–[32]. Li *et al*. [29] proposed a DRL-based vehicular task scheduling algorithm to minimize system cost while prioritizing tasks based on deadlines and dependencies. The authors modeled the scheduling problem as a Markov Decision Process (MDP) to address the challenges associated with the dynamic environment of vehicular networks. Dai *et al*. [30] presented a model for uploading and migrating tasks in an edge-cloud collaborative architecture, with the aim of minimizing the latency of task execution. The authors devised a probabilistic computation offloading approach to optimize this process and achieved optimal results through iterative means. Ma *et al*. [31] developed a solution to address challenges associated with highly dynamic vehicular network environments. The researchers introduced an enhanced heterogeneous earliest finish time algorithm that relies on gradient routing to address the problem of joint optimization of computation offloading and routing. This algorithm aims to maintain alignment between a vehicle's movement direction and the migration direction of tasks during offloading. By efficiently offloading tasks onto edge nodes along the routing path, this approach minimizes migration expenses and optimizes the distribution of computing resources across the network.

However, existing studies have focused mainly on the direct migration of vehicular data, disregarding the issues of data security and handover. This oversight is concerning because if the security of data exchange between vehicular network entities is not effectively ensured, it can pose serious threats to user privacy and even driving safety [33]. To address this critical issue, it is imperative to integrate robust data security measures and seamless handover protocols within the proposed solutions in this space.

B. Blockchain Integration for Secure V2X Services

Blockchain technology, originally conceptualized as the backbone of cryptocurrencies, has rapidly emerged as a versatile tool for secure data sharing and decentralized consensus mechanisms. Its decentralized nature and cryptographic principles make it highly resilient to tampering and fraud, thus gaining popularity beyond its initial applications. With the potential to enhance data security, trust, and transparency, blockchain has gained attention in the field of edge computing and vehicular networks [34]. Blockchain functions as a distributed ledger that records transactions across multiple nodes in a network. Each transaction, or block, is cryptographically linked to the previous one, forming an immutable chain of data blocks. This decentralized architecture eliminates the need for a central authority, mitigating single points of failure and reducing the risk of data manipulation or unauthorized access.

In recent years, researchers have explored the integration of blockchain technology into V2X systems to address security and trust issues inherent in vehicular networks. By leveraging blockchain, V2X services can achieve secure and transparent data sharing among vehicles, infrastructure, and other network entities. In [35], a blockchain-based framework for secure V2X data processing was proposed to address challenges related to efficient energy usage and resource utilization by utilizing edge servers to reduce latency. However, the proposed framework does not consider how the performance of the system could be affected by different types of messages, especially those that have strict latency requirements. This could potentially limit the effectiveness of the framework in certain scenarios. Zhang *et al.* [36] proposed a blockchain-based, hierarchical VEC platform. This approach uses a trust model to secure vehicle communication links, and the blockchain system manages the entire architecture. The aim is to optimize MEC performance while ensuring blockchain consensus. The authors modeled a joint optimization problem as an MDP and proposed a deep compressed neural network scheme to solve it.

Cui *et al*. [37] introduced a blockchain-based VEC platform to improve the efficiency and security of computing. They proposed a centralized controller equipped with a heuristic algorithm to optimize computation delay and implemented the platform in a real-world scenario. Zheng *et al*. [38] proposed a secure computation offloading framework for a blockchain-based vehicular network. The framework consists of a hierarchical architecture for security, a trusted access control mechanism using smart contracts, and a dynamic offloading solution based on DRL. The framework is designed to address the security and privacy challenges of offloading computation tasks to untrusted servers, and it provides a dynamic solution for optimizing offloading decisions and resource allocation. Ren *et al*. [39] presented a two-layer distributed SDN architecture with an integrated VEC platform using blockchain to enhance delay-sensitive applications and reduce energy consumption, achieved through a DRL-based algorithm. To incentivize resource sharing for V2V computation offloading, Shi *et al*. [40] proposed a blockchain-enabled framework using dynamic pricing and DRL. A key innovation is the integration of a dynamic pricing scheme, where the task vehicle pays a service price proportional to the computation size of the selected service vehicle executing the offloaded task. Pricing, along with carefully designed utility functions for both vehicles, provides short-term incentives for

resource contribution. Furthermore, the reliability of vehicles in resource allocation, evaluated from historical offloading transactions recorded on the blockchain, is used for service vehicle selection and consensus node rewards. Vehicles with higher reliability have a higher chance of being selected for offloading and obtaining rewards, thus providing long-term incentives to maintain high reliability. The DRL-based task allocation algorithm is used to dynamically determine the service price and service vehicle to maximize the long-term utility of the task vehicle. This framework combines pricing incentives, DRL-based adaptation, and blockchain-enabled reliability management to incentivize resource sharing for V2V offloading in a secure and reliable manner. Liu *et al.* [19] introduced a blockchain-secured VEC framework for V2V resource trading, aiming to maximize system utility through incentivizing selfish vehicles in a decentralized architecture. Wang *et al.* [18] proposed a consortium blockchain solution to improve security and incentivize resource sharing in VEC. Their approach includes multi-step smart contracts for secure resource sharing and contract-based incentives to maximize utility and social welfare for VEC participants. Lang *et al.* [7] introduced a blockchain-based cooperative computation offloading framework to enhance the security of V2I and V2V computation offloading. Their approach included a combined consensus mechanism for secure information sharing between resource-idle vehicles. The authors also developed a cooperative computation offloading game to validate the effectiveness of their decision-making process.

As it can be seen, the blockchain technology has already been employed for enhancing data security within VEC systems. However, none of the proposed approaches explored blockchain's potential within a versatile, all-encompassing V2X service delivery platform. Furthermore, the aspect of traffic prioritization of V2X applications is not addressed in any existent approaches. This is critical in the quest to obtain low latency and reliable service delivery QoS.

This paper proposes a holistic framework for secure delivery of V2X services by integrating the blockchain with VEC. Our approach guarantees the reliability of services through optimization of a priority and energy-aware utility function.

III. System Model

This section outlines the proposed BEVEC framework architecture and then it presents an analysis of the latencies involved and describes the proposed utility function aimed at achieving reliable and low latency V2X service delivery.

A. BEVEC Description

The BEVEC framework consists of three layers: the vehicle layer, the edge layer, and the blockchain layer, as shown in Fig. 1.

The **vehicle layer** is composed of $\mathcal{V} = \{1, 2, 3, \ldots, V\}$ vehicles, each of which can function as either a V2X service requester or a service provider. In the **edge layer**, multiple BSs and RSUs, denoted as $\mathcal{N} = \{1, 2, 3, \ldots, N\}$, serve dual roles as VEC servers and nodes of the permissioned blockchain. This enables them to function as V2X service providers while

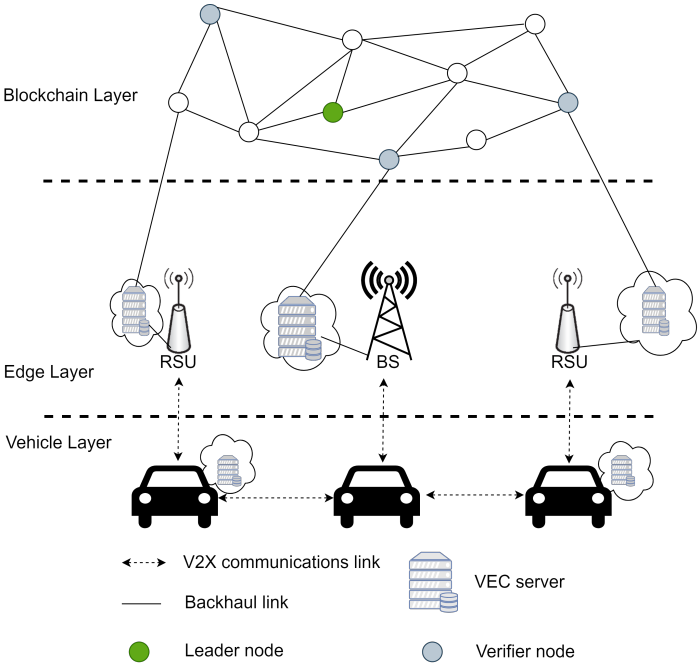

Fig. 1. BEVEC architecture

improving the security and integrity of the entire framework. In the **blockchain layer**, each VEC server serves as a dedicated blockchain node.

Every V2X application is represented in terms of exchange of messages $\mathcal{M}_{ij}$ transmitted from network entity i to entity j. Each message is described as a tuple (c_i, T_i^{max}, ρ_i), that represents application's content, maximum tolerable latency, and traffic priority level, respectively. This definition opens a wide array of possibilities for V2X applications. These possibilities range from enabling vehicles to broadcast decentralized environmental notification messages (DENMs) or cooperative awareness messages (CAMs) to specific applications, such as computation offloading requests or participating in resource sharing. It also simplifies VEC handovers and service migrations and facilitates shared offloading.

The traffic priority and delay sensitivity of $\mathcal{M}_{ij}$ depend on their content. For instance, messages with high priority levels require low latency and high reliability, while messages with low priority levels, such as offloading requests or resource sharing, can tolerate higher latency and are less delay-sensitive.

Leveraging VEC enables efficient offloading of computation-intensive messages, facilitating prompt and reliable V2X service delivery. This not only reduces latency, but also enables real-time data processing and analysis for V2X applications, highlighting the critical role of VEC in ensuring seamless and efficient V2X communication.

1) BEVEC's Dual Layer Verification Process: is ensured by i) a local verification process, on one hand, and ii) the blockchain layer, on the other hand. When a vehicle broadcasts its message, all nearby vehicles, RSUs, and BSs within its communication range will receive it. To ensure privacy and message integrity, only RSUs or BSs are granted access to the message content.

(i) The local verification process initiates when a RSU receives a message from a nearby vehicle that requires accuracy verification. To validate the accuracy of the broadcasted message, at least two other vehicles in proximity must participate by sending confirmation messages. If the local verification process is successful and receives confirmation from at least two other vehicles, indicating the message's accuracy, the message is securely recorded in the blockchain as a valid transaction. If the local verification process fails, the RSU waits until the specified expiration time of the local verification period. If insufficient confirmations are received within this timeframe, the message is disregarded. For unicasted messages, the RSU's role is to transmit message accuracy acknowledgments to nearby vehicles and await their responses. Validation is achieved through the successful receipt of acknowledgments from nearby vehicles, typically requiring acknowledgment from at least two-thirds of the nearby vehicles. If the RSU receives the requisite acknowledgments confirming the message's accuracy, it considers the message a valid transaction and adds it to the blockchain. Conversely, if there is an insufficient number of successful acknowledgments within the designated time frame, the RSU disregards the message.

(ii) The blockchain layer offers each network entity a unique identification consisting of a public key and a private key. This unique identification enables the entities to utilize asymmetric cryptography for secure message transmission. By incorporating these cryptographic techniques, the blockchain layer ensures the confidentiality and integrity of messages exchanged within the network.

A unicasted message originating from the network entity i and intended exclusively for the entity j must be signed using the private key of the entity i, denoted as Sig_i. This signature process is essential to establish the authenticity and integrity of the message. This signing process is as follows.

$$\mathcal{M}_{ij}^{U} = E_{PK_j}(\mathcal{M}_{ij}||ts||Sig_i), \tag{1}$$

where E_{PK_j} indicates that the message $\mathcal{M}_{ij}$ is encrypted using the public key associated with the entity j. Furthermore, the variable ts represents the timestamp, capturing the precise moment when the message was generated.

However, periodic transmission of cooperation information is vital in vehicular networks, where each vehicle must regularly broadcast information such as its current location, speed, direction of movement, and available resources. To achieve this, a broadcast message using a unique key pair, known as the broadcast key pair, is employed for transmission as follows.

$$\mathcal{M}_{ij}^{B} = E_{PK_B}(\mathcal{M}_{ij}||ts||Sig_i), \tag{2}$$

In this configuration, the public key is known to all entities of the network, while the private key is shared among BSs and RSUs. The broadcast key pair enables efficient communication between vehicles and infrastructure elements, ensuring that the necessary data can be securely and consistently shared across the network. Public keys enable vehicles to encrypt their messages so that they can only be decrypted by authorized BSs and RSUs possessing the corresponding private key.

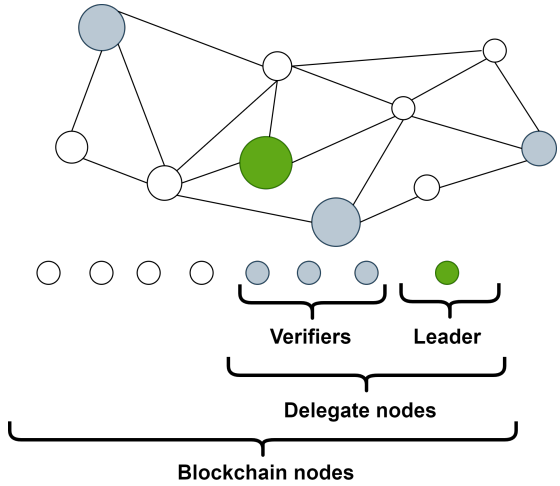

Fig. 2. Delegate node selection: Nodes with larger diameters (higher collected utility) have a greater probability of being chosen.

2) BEVEC's Consensus Mechanism: The security and immutability of data in the blockchain are typically upheld by a consensus mechanism, which is a set of rules that all nodes in the network must follow in order to agree on the state of the ledger. However, this mechanism can introduce delays to the network, as it can take time for all nodes to reach a consensus on a new block. Delegated Proof of Stake (DPoS) is a consensus mechanism that addresses this issue by relying on a voting and selection process to choose a small number of delegates to validate blocks. This reduces the time and energy required to reach consensus, while still securing the blockchain against centralization and malicious activities [10]. The BEVEC framework's blockchain layer consensus mechanism builds on the foundational principles of DPoS and comprises two fundamental elements: delegate selection and block production and verification.

A subset of blockchain nodes denoted $\hat{\mathcal{N}}$ ($\hat{\mathcal{N}} \subset \mathcal{N}$), is chosen as delegates based on their collective utility, as elaborated in the following subsection III-C. In the block production and verification phase, the delegates are divided into a leader node and verifier nodes (Fig. 2). The leader node is responsible for collecting transactions and producing blocks, whereas the verifier nodes focus on verifying blocks. During each block production process, one of the $|\hat{\mathcal{N}}|$ delegates is selected as the leader node in a round-robin fashion, ensuring that each delegate has the opportunity to become a leader and produce blocks. The other delegates act as verifiers. In a specific block production and verification process, the leader collects transactions and computes a correct hash to generate an unverified block. The block verification process follows a three-phase protocol that involves block broadcast, block verification, and confirmation (Fig. 3). During the block broadcast phase, the leader transmits $|\hat{\mathcal{N}}| - 1$ messages to the other delegates for verification. In the block verification phase, each verifier first validates the signature of the received message. Subsequently, verifiers assess the correctness of each transaction and then

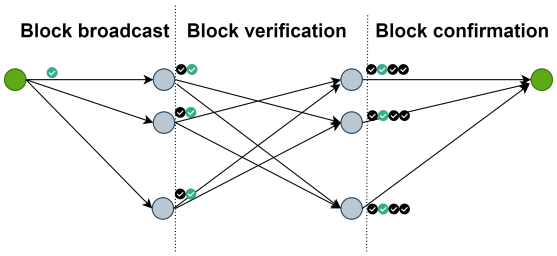

Fig. 3. Block verification process.

share their signed audit results in a distributed manner. In the confirmation phase, each verifier compares its audit result with those received from other verifiers and sends a confirmation message to the leader. The confirmation message includes the audit results and comparison outcomes. Upon receiving confirmation from all delegates, the leader analyzes them to determine the block's correctness. If more than two-thirds of verifiers agree on the block's validity, the leader broadcasts it to all delegates for storage. Nodes that are not included in the delegate commission periodically synchronize with nearby delegates to obtain the latest blockchain. If all delegates have served as leaders once, their order is shuffled, and they produce future blocks in a round-robin manner once again. In cases where a delegate fails to create a block during their turn, the block is skipped and transactions from the skipped block are transferred to the next one.

Note that the BEVEC framework's dual layer mechanism comes with a cost. This cost is due to the communication overhead imposed by the local verification process and the blockchain's consensus mechanism and it is reflected in the next section on *Latency Analysis*. However, the dual-layer mechanism is required to provide an additional level of security and reliability.

B. Latency Analysis

Fig. 4 illustrates the steps involved in sending a message from vehicle i to the network's entity j, processing it, and storing it on the blockchain in the BEVEC framework.

The overall delay can be expressed as follows:

$$T_{\mathcal{M}_{ij}} = \tau_{ij} + \tau_j^{proc} + \tau_j^{verf} + x_j \tau_j^{bv} + (1 - x_j) T_{\mathcal{M}_{jj\prime}}, \quad (3)$$

where τ_{ij} denotes the transmission latency for sending message $\mathcal{M}_{ij}$ from the entity i to j. τ_j^{proc} and τ_j^{verf} are respectively the processing and verification delays incurred in the j-th entity and τ_j^{bv} denotes the block verification latency of the blockchain. $x_j \in \{0, 1\}$ is a binary variable indicating whether the entity j is a blockchain node or not. The processed message $\mathcal{M}_{ij}$ must be recorded as a transaction in the blockchain before it can be considered complete. However, when the entity j is a vehicle or in VEC handover, it should send the processing result to the nearest blockchain node in the form of a new message $\mathcal{M}_{jj\prime}$. The new message should contain the signature of the origin entity i with revised maximum tolerable latency. The revised value can be expressed as $T_j^{max} = T_i^{max} - (\tau_{ij} + \tau_j^{proc})$. To ensure that the message $\mathcal{M}_{jj\prime}$ is integral to the vehicle i, entity $j\prime$ (i.e. the blockchain node) must verify the integrity of the message from entity j with respect to entity i.

The transmission latency can be expressed as follows [40]:

$$\tau_{ij} = \frac{\mathcal{S}(\mathcal{M}_{ij})}{R_{ij}}, \quad (4)$$

where $\mathcal{S}(\mathcal{M}_{ij})$ is the size of the message $\mathcal{M}_{ij}$ in bits and R_{ij} denotes the communication data rate between the entities of the vehicular network i-th and j-th, which can be expressed as the Shannon-Hartley capacity (5) or the finite length capacity (7) depending on the size of the message $\mathcal{S}(\mathcal{M}_{ij})$. In cases where $\mathcal{S}(\mathcal{M}_{ij})$ is very large, the decoding error rate is very small. However, in the majority of V2X applications, the data exchanged is typically minimal, increasing the likelihood of encountering a non-zero decoding error. As a result, Shannon-Hartley capacity is not applicable, as the achievable rate falls within the regime of finite block-length channel coding [41], [42]. For messages of sufficient size, the communication rate is defined as follows:

$$R_{ij}^{\infty} = B \log_2(1 + \Gamma_{ij}), \quad (5)$$

where B is the channel bandwidth, the signal-to-noise ratio (SNR) between the entities i-th and j-th of a vehicular network is denoted by Γ_{ij} and can be expressed by the following equation:

$$\Gamma_{ij} = \frac{p_{ij} |h_{ij}|^2 d_{ij}^{-\nu}}{\sigma^2}, \quad (6)$$

where p_{ij} is the transmission power of i-th vehicular network's entity toward the j-th element of the network, h_{ij} and d_{ij} are presenting the Rayleigh fading coefficient and distance, respectively, ν denotes the path loss exponent, and σ^2 is the noise power. The data rate for short-length messages is represented as [42]:

$$R_{ij} = R_{ij}^{\infty} - \sqrt{\frac{U_{ij}}{\mathcal{S}(\mathcal{M}_{ij})}} Q^{-1}(\epsilon), \quad (7)$$

where $Q^{-1}(.)$ is the inverse of the Gaussian Q-function as (8), $\epsilon > 0$ is the transmission error probability, and U_{ij} represents the characteristic of the channel called the channel dispersion, i.e., U_{ij} determines the stochastic variability of the channel when compared to a deterministic channel with the same capacity, given by (9).

$$Q(x) = \frac{1}{2\pi} \int_x^{\infty} e^{-\frac{t^2}{2}} dt, \quad (8)$$

$$U_{ij} = 1 - \frac{1}{(1 + \Gamma_{ij})^2}, \quad (9)$$

We define $P_{ij} = \frac{f_{j,d}}{\Omega_i}$ as the CPU cycles (Hz) required to process the message $\mathcal{M}_{ij}$. Here, Ω_i represents the number of CPU cycles needed to process each bit of message $\mathcal{M}_{ij}$, which is appended to the content section of every message. Furthermore, each VEC server is equipped with D_j virtual machines based on its computing capabilities. The variable $f_{j,d}$ represents the computational capacity of a single virtual

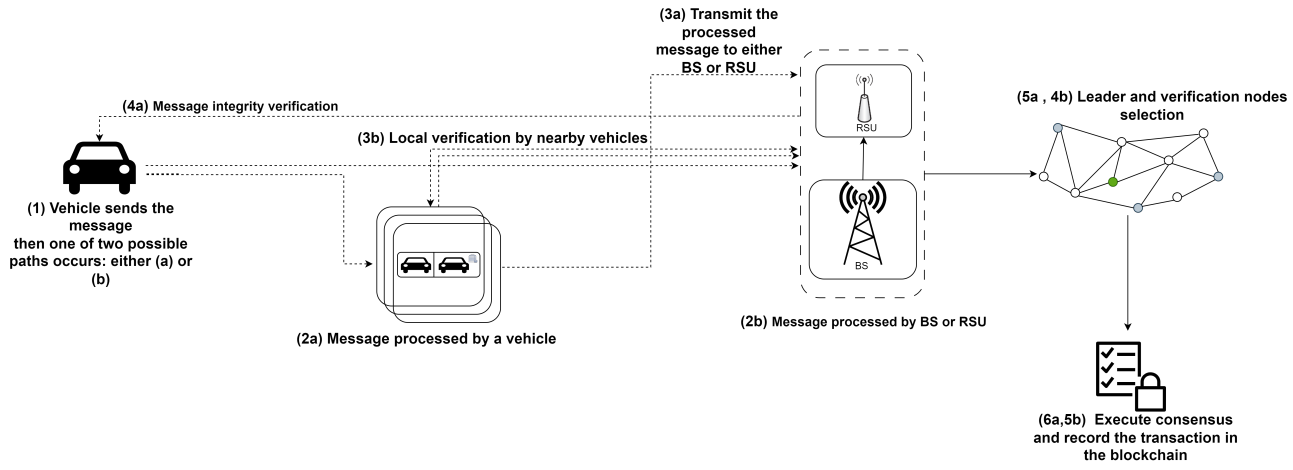

Fig. 4. Life cycle of processing a message in BEVEC

machine within entity j. The processing delay can then be computed as follows [8]:

$$\tau_j^{proc} = \frac{\mathcal{S}(\mathcal{M}_{ij})}{P_{ij}}, \tag{10}$$

τ_j^{verf} can be expanded by considering the set of $\mathcal{K} = \{1, 2, 3, \ldots, K\}$ as the local verification vehicles and using a predefined time as the timeout time for the local verification phase as follows:

$$\tau_j^{verf} = \min\Big(T_{timeout_j}, 2 \times \max\{\tau_{j1}, \tau_{j2}, \ldots, \tau_{jK}\}, \max\{\tau_1^{proc}, \tau_2^{proc}, \ldots, \tau_K^{proc}\}\Big), \tag{11}$$

The value of $T_{timeout_j} \in [0, T_{max}]$ varies depending on whether the message requires local verification or not. For instance, low-priority messages are not subject to local verification. Therefore, by setting the $T_{timeout_j}$ equal to zero, the τ_j^{verf} term in the overall delay of (3) will automatically be removed. It is worth noting that, for the sake of simplicity, the transmission latency for sending the acknowledge request message and receiving it has been assumed to be equal in (11).

τ_j^{bv} consists of three parts: 1) block broadcasting, 2) cross-verification among verifiers, 3) block confirm, which can be described similarly to [10] as:

$$\tau_j^{bv} = \tau_j^{bb} + \tau_j^{cv} + \tau_j^{bc}, \tag{12}$$

Denote $j \in \hat{\mathcal{N}}$ as the leader node and $j\prime \in \hat{\mathcal{N}}\backslash\{j\}$ as verifiers. Since the leader j broadcasts its produced block as message $\mathcal{M}_{jj\prime}$ to verifiers simultaneously, the block broadcasting time is determined by the longest block transmission time, so

$$\tau_j^{bb} = \max_{j\prime \in \hat{\mathcal{N}}\backslash\{j\}} \tau_{jj\prime}, \tag{13}$$

where $\tau_{jj\prime}$ is the transmission delay between the leader j and verifier $j\prime$. Cross-verification consists of three steps. Each verifier first performs verification individually to verify the raw block from the leader ($\mathcal{M}_{jj\prime}$) and then broadcasts its verified result to other verifiers. After receiving the verified result ($\mathcal{M}_{j''j\prime}$), verifiers perform a second audit. The block cross-verification time consumption can be written as in (14):

$$\tau_j^{cv} = \max_{j\prime, j'' \in \hat{\mathcal{N}}\backslash\{j\}, j\prime \neq j''} \{\tau_{j''j\prime} + \tau_{j\prime}^{proc}(\mathcal{M}_{jj\prime}) + \tau_{j\prime}^{proc}(\mathcal{M}_{j''j\prime})\}, \tag{14}$$

Block confirm time is determined by the longest second-audit result transmission time, which is:

$$\tau_j^{bc} = \max_{j\prime \in \hat{\mathcal{N}}\backslash\{j\}} \tau_{j\prime j}, \tag{15}$$

Overall energy consumption is determined by the combined effect of transmission, processing, local verification, and the consumed energy in the blockchain layer. Assuming that the power consumed during each stage of the process, local verification, and blockchain is identical, denoted as p_0, the total energy consumed can be expressed as:

$$E_{ij} = p_{ij}\tau_{ij} + p_0\left(\tau_j^{proc} + \tau_j^{verf} + x_j\tau_j^{bv}\right), \tag{16}$$

C. Proposed Utility Function and Problem Formulation

The core objective of the BEVEC framework is to minimize energy consumption while maintaining security and privacy during data exchange. However, achieving this objective requires the implementation of an incentive mechanism, particularly for participants within the vehicle layer. The utility function defined to meet the aforementioned objective is:

$$\mathbb{I}_{ij}(E_{ij}, \mathcal{S}(\mathcal{M}_{ij}), \rho_{ij}) = \left(\omega_1|I_{ij}^E|^2 + \omega_2|I_{ij}^S|^2 + \omega_3|I_{ij}^\rho|^2\right)^{\frac{1}{2}}, \tag{17}$$

where I_{ij}^E, I_{ij}^S, and I_{ij}^ρ represent the energy, message, and priority utilities, respectively. The weight factors ω_1, ω_2 and ω_3 are used to balance the effects of these utilities where $\omega_1 + \omega_2 + \omega_3 = 1$.

The utility functions are extended in the following manner:

$$I_{ij}^E = e^{a_E(\frac{E_{th} - E_{ij}}{E_{th}})}, \tag{18}$$

$$I_{ij}^S = e^{a_S(\frac{S_{ij} - S_{th}}{S_{th}})}, \tag{19}$$

$$I_{ij}^\rho = e^{a_\rho(\frac{\rho_{ij} - \rho_{th}}{\rho_{th}})}, \tag{20}$$

where E_{th}, S_{th}, and ρ_{th} represent the thresholds for demanded energy, message size, and priority level, respectively, and $a_E > 0$, $a_S > 0$, and $a_\rho > 0$ denote the weighted factors for the utility function. If the energy value exceeds the threshold while the message size and priority level are below the specified thresholds, the utility function values are always greater than 0 and less than 1, meaning $0 \leq I_{ij}^E \leq 1$, $0 \leq I_{ij}^S \leq 1$, and $0 \leq I_{ij}^\rho \leq 1$. On the other hand, if the energy is below the threshold but the message size and priority level requirements are satisfied, the utility function values are always greater than or equal to 1, namely $1 \leq I_{ij}^E$, $1 \leq I_{ij}^S$, and $1 \leq I_{ij}^\rho$. Furthermore, the utility function value decreases as energy consumption increases, while the message size and priority level utility function values increase as the message size and priority level increase. In this case, the value of $\mathbb{I}_{ij}$ increases as energy consumption decreases for sufficient message size and reasonable priority levels.

The problem of utility maximization can be expressed as:

$$\text{P1}: \quad \max \sum_{i\in\mathcal{P}} \sum_{j\in\mathcal{P}\setminus\{i\}} \alpha_{ij}\mathbb{I}_{ij}, \tag{21a}$$

$$\text{s.t.} \quad \text{C1}: \sum_{j\in\mathcal{P}\setminus\{i\}} T_{\mathcal{M}_{ij}} \leq T_i^{max}, \quad \forall i \in \mathcal{P} \tag{21b}$$

$$\text{C2}: \alpha_{ij} \in \{0,1\}, \quad \forall i \in \mathcal{P}, \forall j \in \mathcal{P}\setminus\{j\} \tag{21c}$$

where $\mathcal{P} = \mathcal{V} \cup \mathcal{N}$ is the set of all entities in the vehicle and edge layer, binary variable α_{ij}, indicates the existence of a connection between entity i and j, the constraint (21b) is enforced to ensure that transmitted messages are processed within the maximum tolerable time.

Problem P1 illustrates the interaction between the utility function and performance metrics of the BEVEC. Utility functions are inversely proportional to energy consumption, which means that higher energy consumption leads to lower utility. Additionally, condition C1 specifies that messages must be delivered within a specific timeframe to receive the associated utility. As a result of this requirement, low latency is preferred. In addition to latency reduction, meeting this deadline increases the probability of successful message delivery, since lower energy consumption naturally results in lower latency.

IV. Proposed DRL approach

Problem P1 presents a challenge due to the binary variable α_{ij}, which makes the feasible set and the objective function non-convex. While approximate algorithms can be used to solve the problem, they may not scale well, particularly as the number of vehicles increases. Moreover, the ad-hoc and on-demand nature of messages exchanged in the vehicular network introduces additional complexities. Conventional optimization methods may struggle to keep up with the changing demand over time, especially as the network lacks a fixed infrastructure.

Given the challenges associated with decision-making in problem P1, it is essential to handle high-dimensional and time-varying features appropriately. However, conventional models-based algorithms, such as greedy and meta-heuristic algorithms, are not suited for scaling up in large applications due to the requirement for near-complete information [43]. To overcome scalability and adaptability limitations, an efficient model-free solution based on DRL is proposed.

Before exploring the proposed approach, it is essential to establish the RL-compatible version of the problem, which encompasses the state and action spaces and the reward function. In the following, the set $\mathcal{T}$ represents the discrete time intervals of the system.

A. State

The state space is the reflection of the observed vehicular environment. Let $\{\mathcal{S}, t \in \mathcal{T}\}$ be the state space. The state in time period t evolves across $\mathcal{T}$ and can be expressed as:

$$s_t = \{\mathbf{M}(t), \mathbf{R}(t), \mathbf{F}(t), \mathbf{\Lambda}(t)\}, s_t \in \mathcal{S}, \tag{22}$$

where

- $\mathbf{M}(t)$ is the flattened vector of $|\mathcal{P}| \times |\mathcal{P}-1|$ message matrix, representing the set of message pairs at time period t;
- $\mathbf{R}(t)$ is the rate vector with the same dimensions as the message vector, representing communication data rates between different entities of the vehicular network at time period t;
- $\mathbf{F}(t)$ is a vector that represents the current processing capability of each entity at time period t;
- $\mathbf{\Lambda}(t)$ is a $|\mathcal{P}| \times |\mathcal{P}-1|$ flattened matrix and contains elements $\lambda_{ij} \in \{-1, 0, 1\}$. These elements represent the relative movement between vehicles and other network's entities at time period t, where a value of 1 indicates that they are approaching each other, -1 indicates that they are moving away from each other, and 0 indicates that their positions are relatively fixed.

B. Action

Let $\{\mathcal{A}, t \in \mathcal{T}\}$ denote the action space. We define the action vector $a_t \in \mathcal{A}$ at time period t as follows:

$$a_t = \text{vec}\left(\begin{bmatrix} \alpha_{12}, \alpha_{13} & \cdots & \alpha_{1P} \\ \vdots & \ddots & \vdots \\ \alpha_{P1}, \alpha_{P2} & \cdots & \alpha_{P(P-1)} \end{bmatrix}^T \right), \tag{23}$$

The objective of problem P1 is to find the appropriate values of α_{ij} that maximize the overall utility function.

C. Reward Function

Once the action a_t is taken, the environment will provide an immediate reward, which can be expressed as follows:

$$\psi_t(s_t, a_t) = \begin{cases} \mathbb{I}_{ij}, & \text{if C1} \cap \text{C2} \\ -\Upsilon, & \text{otherwise} \end{cases} \tag{24}$$

where $\Upsilon > 0$ is the constant that represents the penalty.

The optimal solution of problem P1 is achieved by maximizing the expected cumulative discounted rewards also called the long-term reward and defined in (25).

$$\Psi = \max \mathbb{E}\Big[\sum_{t=0}^{T} \gamma^t \psi_t(s_t, a_t)\Big], \tag{25}$$

where $\gamma \in [0,1]$ is the discount factor which indicates the weight of the future reward. For a fixed t, a larger value of γ corresponds to a greater emphasis on the future reward. It is worth noting that, when γ is fixed, the quantity γ^t approaches zero as t becomes sufficiently large. This implies that the contribution of the future reward to the long-term reward decreases over time.

D. A Novel DRL-based Approach

We begin by providing a concise overview of the foundational principles of DRL, with a particular focus on Q-learning algorithms. This overview lays the groundwork for our proposed methodology, which we introduce subsequently to address the problem P1.

1) Technical background: DRL is a powerful approach that combines RL with deep learning to tackle problems with high-dimensional raw data inputs [44]. In the training process of DRL, a deep neural network called the Deep Q-Network (DQN) is used to approximate the action-state pair and the Q function $Q(s,a;\theta)$, where θ represents the weights of the neural network. The DQN is trained iteratively by updating its weights θ to approximate the real Q values. Two techniques, experience buffer and target network with target Q function $Q_T(s,a;\theta_T)$, are employed to improve the training efficiency of DQN. To train the main DQN, a mini-batch of size M experiences $(s_t^{(i)}, a_t^{(i)}, \psi_t^{(i)}, s_{t+1}^{(i)})$ stored in the buffer is used to minimize the loss which can be expressed as:

$$L(\theta_t) = \frac{1}{M}\sum_{i=1}^{M}(y_T^{(i)} - Q(s_t^{(i)}, a_t^{(i)};\theta_t))^2, \tag{26}$$

The loss is defined as the squared difference between the target Q value, $y_T^{(i)}$, and the estimated Q value, $Q(s_t^{(i)}, a_t^{(i)};\theta_t)$, which is also known as the temporal difference error (TDE). The target Q value is given by:

$$y_T^{(i)} = \psi_t^{(i)} + \gamma \max_{a_{t+1}} Q_T(s_{t+1}^{(i)}, a_{t+1}^{(i)};\theta_{T_t}), \tag{27}$$

The parameters of the DQN network are updated using the gradient descent method with a learning rate, lr, as shown in the following equation:

$$\theta = \theta + lr \times \frac{1}{2}\nabla_\theta (L(\theta))^2, \tag{28}$$

Additionally, the target DQN network parameters can be updated periodically every G step using the:

$$\theta_{T_t} = \theta_{t-G}, \tag{29}$$

*2) **3DPER** - a novel double-dueling DQN with prioritized replay experiences:* To solve problem P1 and find optimal matches between different entities, a double-dueling DQN with prioritized replay experiences (3DPER) is proposed. To manage extensive action space of (23), $|\mathcal{P}|$ parallel Q networks are employed, as illustrated in Fig. 5. Adopting the strategy from Wang et al.'s dueling network approach [45], and considering that not all actions impact the state in certain scenarios, 3DPER separates each of the Q-values into two components: the state's value ($V(s)$) and the advantage of executing a specific action within that state ($A(s,a)$). Thus, the Q-value specified for entity j can be represented as a combination of these two streams:

$$Q_j(s, a_j;\theta_j) = V_j(s) + A_j(s, a_j), \tag{30}$$

where a_j denotes an action applicable to entity j, which can also be interpreted as the j-th row of the action vector before its vectorization, i.e. $a = \mathrm{vec}([a_1, \ldots, a_j, \ldots]^T)$.

Define $Q = \sum_{j=1}^{|\mathcal{P}|} Q_j$; then, the loss function in (26) can be replace by:

$$L(\theta_t) = \frac{1}{M}\sum_{i=1}^{M}\Big[(\psi^{(i)} + \gamma\Big(\max_{a\prime}\sum_{j=1}^{|\mathcal{P}|} Q_{T_j}(s\prime, a\prime_j)\Big) - Q(s,a))^2\Big], \tag{31}$$

where $s\prime$ and $a\prime_j$ are equal to s_{t+1} and $a_{j_{t+1}}$ respectively. Since $\max_{a\prime}\sum_{j=1}^{|\mathcal{P}|} Q_{T_j}(s\prime, a\prime_j))$ is equal to $\sum_{j=1}^{|\mathcal{P}|}\max_{a\prime_j} Q_{T_j}(s\prime, a\prime_j))$ and due to quality of $\max_{a\prime_j} Q_j(s\prime, a\prime_j)$ with $Q_j(s\prime, \arg\max_{a\prime_j} Q_j(s\prime, a\prime_j))$ and for decoupling action selection from action evaluation to avoid overestimation and reduce biases, as described in [46], the final form of the loss function in 3DPER can be written as:

$$L(\theta_t) = \frac{1}{M}\sum_{i=1}^{M}\Big[(\psi^{(i)} + \gamma\Big(\sum_{j=1}^{|\mathcal{P}|} Q_{T_j}(s\prime, \arg\max_{a\prime_j} Q_j(s\prime, a\prime_j))\Big) - Q(s,a))^2\Big], \tag{32}$$

Furthermore, 3DPER utilizes a replay memory experience buffer to store and randomly sample experiences, but with a distinctive twist. Unlike traditional replay memory, 3DPER assigns a priority level to each experience based on the magnitude of the TDE. This prioritization scheme allows for the sampling of experiences that have a greater impact on the learning process, leading to faster convergence and improved performance, as demonstrated in [47]. The complete approach of 3DPER is described in Algorithm 1.

E. Complexity Analysis

1) Computation complexity: In this subsection, we analyze the computational complexity of the proposed 3DPER algorithm and compare it with the conventional DQN algorithm.

The dimensionality of the action space is a critical factor in determining the computational complexity of DQN algorithms. In conventional DQN, each entity has $|\mathcal{P}|-1$ potential actions, resulting in an exponential increase in the action space dimension with the number of entities, i.e., $(|\mathcal{P}|-1)^{|\mathcal{P}|}$. This makes it intractable for larger values of $|\mathcal{P}|$. In contrast, the 3DPER architecture proposed in this paper aggregates the Q network of each entity, resulting in a linear increase in the action space dimension with the number of entities, i.e., $(|\mathcal{P}|-1)|\mathcal{P}|$. This significantly reduces the computational complexity of the algorithm, making it more scalable and practical for larger problems.

The state s_t in the 3DPER algorithm has $|\mathcal{P}|(3|\mathcal{P}|-2)$ parameters, as given by (22). For approximating state-action values using $|\mathcal{P}|$ separate Q networks, each with $\mathcal{L}$ fully connected hidden layers along with value and advantage streams containing W_l parameters, the overall computation complexity of the 3DPER algorithm in

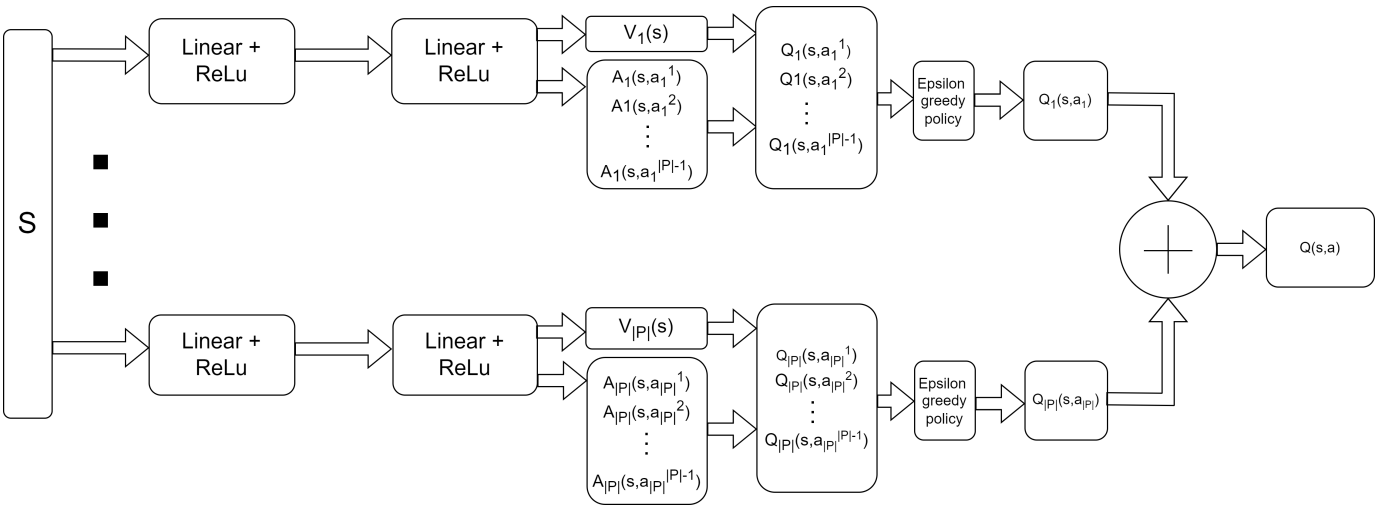

Fig. 5. 3DPER architecture.

Algorithm 1 3DPER Algorithm

Input: experience buffer $\mathcal{B}$, minibatch size M, initial weights θ and θ_T, minimum exploration probability $\varepsilon_{\min}$, exploration decay rate $\delta\varepsilon$, discount factor γ, learning rate lr;

Initialize the experience replay buffer $\mathcal{B} = \{\}$;
Get the initial state s_0 from environment and set $\varepsilon = 0.99$;

for each episode **do**
  Setup vehicular environment;
  for $t = 1 : T$ **do**
    for $j \in \mathcal{P}$ **do**
      Initialize the main network with random weights θ_j;
      Initialize the target network with weights such that $\theta_{T_j} = \theta_j$;
      #Epsilon greedy policy:
      Choose a random probability p;
      if $p > \varepsilon$ **then**
        $a_{j_t} = \text{argmax}_{a_{j_t}} Q_j(s_t, a_{j_t}; \theta_t)$;
      else
        Randomly select an action a_{j_t};
      end if
    end for
    Update exploration probability ε using

$$\varepsilon = \max\{\varepsilon_{\min}, \varepsilon(1 - \delta\varepsilon)\}$$

    Execute action a_t by aggregating all a_js, and observe the reward r_t and the next state s_{t+1};
    Store the experience (s_t, a_t, r_t, s_{t+1}) in $\mathcal{B}$ with probability P using the method of [47];
    Sample a mini-batch of M experiences $(s_i, a_i, \psi_i, s\prime_i)$ from the experience buffer $\mathcal{B}$ using the importance sampling method of [47];
    Perform a stochastic gradient descent step on $L(\theta)$ in (32);
    Every G steps reset $\theta_T = \theta$;
  end for
end for

terms of the number of required multiplications is given by: $|\mathcal{P}|\left(|\mathcal{P}|(3|\mathcal{P}|-2)W_1 + 2\sum_{l=1}^{\mathcal{L}-1} W_l W_{l+1} + W_{\mathcal{L}}|\mathcal{P}|(|\mathcal{P}|-1)\right)$.

The third term, $W_{\mathcal{L}}|\mathcal{P}|(|\mathcal{P}|-1)$, depends on the action space dimensionality, which is typically increasing exponentially with the number of entities in conventional DQN methods. However, in 3DPER, the action space dimensionality grows linearly with the number of entities, making the third term much smaller than in conventional DQN. This is one of the key advantages of 3DPER, as it makes the algorithm much more scalable and practical for large-scale problems.

2) Communication complexity: In this subsection, we examine the communication complexity of 3DPER and contrast it with traditional DQN algorithms. Because single-agent DQN algorithms have minimal communication complexity [48] and 3DPER is a DQN-based single-agent algorithm, the communication complexity of 3DPER is also minimal and is determined by the size of the state vector, which is on the order of $|\mathcal{P}| \times \bar{S}$ bits, where $\bar{S}$ is the average size of the exchanged messages.

V. Performance Evaluation

A. Experimental Setups

In our experimental setup, we integrated the BEVEC framework into a simulated environment covering an area of approximately 4.48km^2 using the SUMO [1]simulation platform, as illustrated in Fig. 6. This choice of simulation platform is notable for its ability to closely emulate real-world traffic scenarios, providing a robust foundation for our evaluations. Within this simulated area, we positioned 9 BSs and 20 RSUs to mimic a realistic deployment scenario. Each BS has a coverage area of approximately 0.5 km^2, and within these coverage areas, we included approximately 2 RSUs per BS to facilitate comprehensive network coverage and connectivity. This configuration was designed to closely resemble the deployment characteristics observed in practical, urban settings. Furthermore, to assess the performance of the BEVEC framework comprehensively, we introduced varying traffic densities into the simulation environment, enabling

[1]SUMO website: https://eclipse.dev/sumo

us to evaluate its adaptability and efficiency under diverse conditions. To validate the effectiveness and advantages of our proposed 3DPER algorithm, we utilized RLlib and PyTorch and executed the experiments on an Ubuntu 20.04.5 LTS operating system. RLlib leverages Ray to enable distributed processing, harnessing the power of distributed computing resources for faster and more efficient reinforcement learning training [49]. In contrast to Q-learning algorithms, where mathematical analysis of convergence during training is possible, understanding convergence in DQN methods, including ours, is more challenging [44]. Since a theoretical analysis of convergence in DQN algorithms is not feasible, we relied on simulations to evaluate the convergence of our method and ensure its reliability and efficiency. While this approach lacks the theoretical foundation of Q-learning analysis, it provides valuable insights into the practical convergence behavior of our DQN algorithm.

Furthermore, we conducted an in-depth analysis of the 3DPER algorithm's performance across different parameter configurations, examining reward, latency, success rate, and energy consumption. This comprehensive evaluation highlighted the algorithm's advantages and identified optimal parameter settings. To further substantiate our approach, we compared it against other strategies, including:

- Random Method: In this approach, actions are selected entirely at random, leading to a purely exploratory methodology that does not take into account past rewards or environment knowledge.
- Greedy Method: Building upon our previous work [3], this method models network entities as graph nodes and assigns weights to each vertex based on rewarded utility and feasible paths. A well-known shortest path algorithm is then employed to match different network entities.
- DDPG Method: Due to the large action space size of the equivalent MDP for problem P1, the Deep Deterministic Policy Gradient (DDPG) method was used [50]. DDPG uses both an actor and a critic neural network. The actor network learns the policy (the action selection), while the critic network learns the value function to evaluate the chosen actions. As DDPG is capable of handling continuous action spaces only, a rounding method was employed to convert continuous actions into discrete ones to make DDPG applicable.

Table II summarizes detailed information regarding the specific parameter values utilized in our experiments, motivated by [7], [8], [40]. In cases where the specific value of a parameter is not specified, the values are selected uniformly.

TABLE II
SIMULATION PARAMETER TABLE

Parameter	Value
B	10 MHz
f_j (Vehicle)	1 GHz
f_j (RSU)	2 GHz
f_j (BS)	3 GHz
D_j (Vehicle)	1
D_j (RSU)	4
D_j (BS)	8
p_{ij}	0.2 W
p_0	2 W
$\mathcal{S}_{ij}$	[0.0012 - 4] Mbits
Message arrival rate λ	10 messages/s
$\lvert\hat{\mathcal{N}}\rvert$	5
T_i^{max}	{0.25,0.5,1,2,4} s
Ω_i	{5,15,20,25} cycles/bit
σ^2	10^{-14} W
ν	3
γ	0.95
$\varepsilon_{\min}$	0.01
$\delta\varepsilon$	0.005
M	64
Size of $\mathcal{B}$	100000
Υ	100
Optimizer	Adam
Activation function	ReLu

Fig. 6. Simulation environment

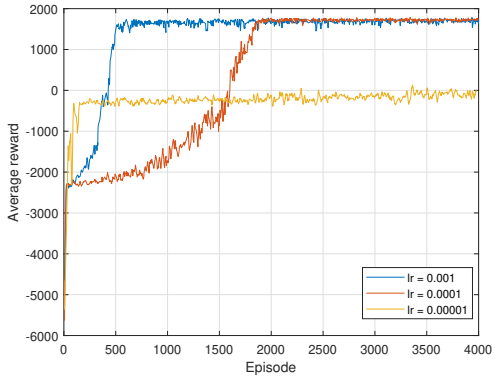

Fig. 7. Impact of different learning rates in convergence of 3DPER algorithm.

B. Results and Analysis

We start by evaluating the 3DPER algorithm's convergence behavior for different learning rates. Fig. 7 shows that different learning rates produce different reward trajectories. For example, with a learning rate of 0.001, the algorithm converges to the maximum reward in about 500 episodes. With a learning rate of 0.0001, the algorithm converges to the maximum reward, but it takes about 2,000 episodes to do so. Notably, with a learning rate of 0.00001, the algorithm fluctuates in the early episodes and eventually gets stuck at a local maximum, which is not the best possible reward.

Because the 0.001 learning rate allows the algorithm to achieve the maximum reward while converging more quickly, we use it for subsequent simulations.

To evaluate how varying numbers of vehicles impact BEVEC performance, we initiate our analysis by comparing reward curves using different algorithms, as illustrated in Fig.

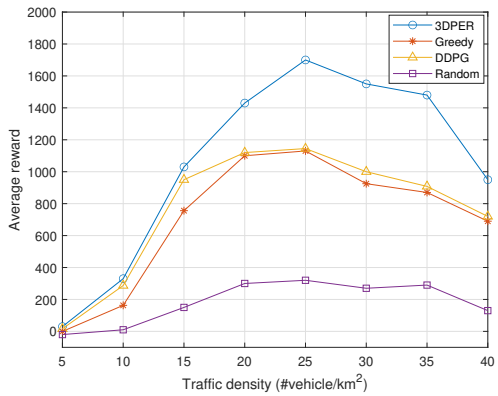

Fig. 8. Average reward under different traffic densities.

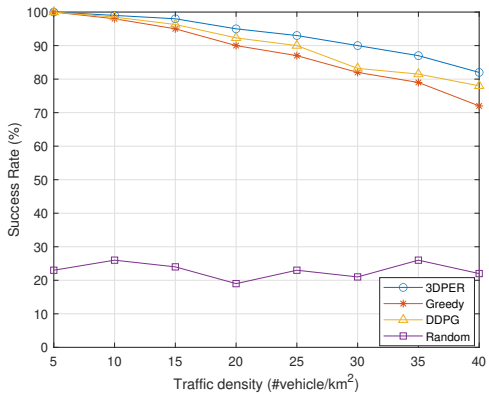

Fig. 9. Success rate under different traffic densities.

8. The comparison highlights that our proposed algorithm outperforms other algorithms in terms of average reward due to several key factors. Unlike the greedy method, which focuses only on maximizing immediate reward, our algorithm optimizes for expected long-term rewards. Additionally, unlike the random algorithm, which explores actions without a specific policy, both our algorithm and DDPG leverage learned policies, leading to higher average rewards. Furthermore, unlike DDPG, which approximates action selection via an actor network that may introduce errors, our approach directly explores the real action space, exploiting the most effective actions. It's worth noting that the average reward exhibits an increasing trend as traffic density rises. However, as the vehicular network's resources become saturated, the average reward begins to decline. Fig. 9 assesses the success rate (SR) of various algorithms across different levels of traffic density. In line with our expectations based on the average reward analysis, the 3DPER algorithm consistently outperforms the other algorithms, with an average improvement of 38% compared to the other methods. This observation highlights a crucial distinction: while in some scenarios, the average rewards of different algorithms may be relatively similar, the actual number of messages exchanged and successfully written to the blockchain within an acceptable time frame varies significantly.

To illustrate, consider the performance of the random algorithm, which achieves an SR of less than 30%. This result indicates that a substantial portion of the messages fail to find the opportunity for timely exchange. Meanwhile, by ignoring the random method, the average improvement of 3DPER to the average of DDPG and greedy algorithms is 4.5%, highlighting the significant advantage of 3DPER in terms of successful message exchange.

Fig. 10 presents the average latency across various traffic densities. Our proposed algorithm stands out for its lower average latency, which is on average 18% lower than other algorithms. Although the latency of greedy and DDPG algorithms is similar to our proposed algorithm, 3DPER still achieves a 2.3% average latency reduction compared to them. This proximity in latency values is because average latency calculations are contingent on successful message deliveries. However, the performance of the random algorithm underscores the significance of prerequisite matching among network entities, which can substantially influence latency. For instance, Fig. 11 portrays the average latency when the maximum tolerable latency for messages is fixed. The average latency remains below the maximum tolerable latency. However, this consideration alone, without factoring in the SR, is insufficient. As we are bound by the constraint of ensuring that the sum of latencies remains below a threshold, the impact of latency must be evaluated in conjunction with SR. Fig. 12 illustrates the SR of various algorithms under varying, yet fixed, maximum tolerable delays. Predictably, the random algorithm records the lowest SR, while the 3DPER algorithm consistently outperforms the greedy and DDPG algorithms in this regard. As the maximum tolerable delay is increased, more time is available for message transmission and processing, which results in improved SR. Fig. 13 examines the average latency under varying exchanged data sizes, where the maximum tolerable latency for each message has been set proportionally. Notably, the 3DPER algorithm exhibits the lowest latency, particularly as the data size increases. This can be attributed to the algorithm's ability to collect more rewards, which in turn allows for a better balance between message size and overall latency, as illustrated in Fig. 8.

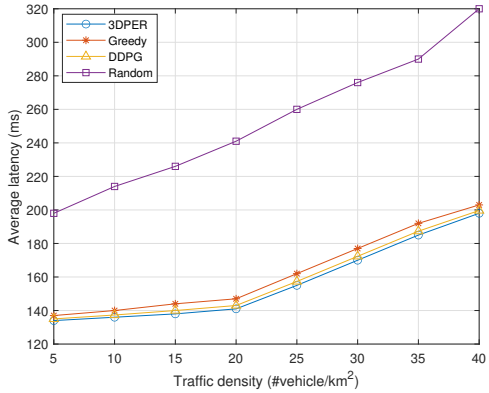

Fig. 10. Average latency under different traffic densities.

Fig. 14 illustrates the average energy consumption trends under varying traffic densities. Notably, the random algorithm exhibits suboptimal performance and excessive energy consumption compared to the other algorithms. As traffic density increases, the energy usage in the other algorithms increases

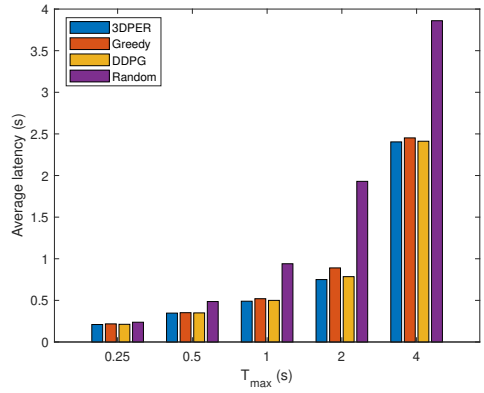

Fig. 11. Average latency versus different $\mathrm{T_{max}}$.

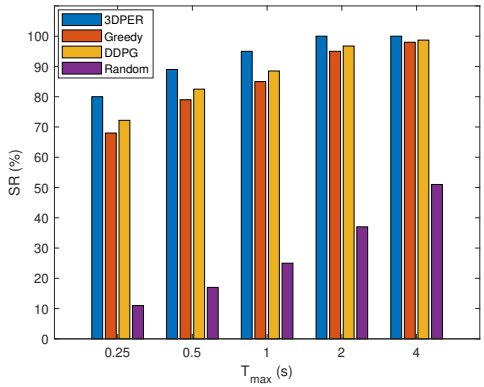

Fig. 12. SR versus different $\mathrm{T_{max}}$.

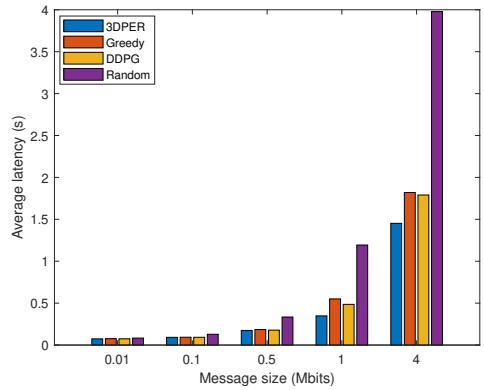

Fig. 13. Average latency versus different message size.

marginally, whereas the 3DPER algorithm demonstrates better performance and reduced energy consumption compared to the greedy and DDPG algorithms. On average, 3DPER reduces energy consumption by approximately 65% compared to other algorithms. By excluding the random algorithm due to its poor performance, 3DPER achieves an average energy reduction of 7.5% when compared to the greedy and DDPG algorithms.

Fig. 15 illustrates how the reward varies in response to changes in the message arrival rate for different schemes. Notably, as the rate increases, the rewards gradually decline and eventually reach a saturation point. This decline is primarily attributed to factors such as extended processing times and execution failures resulting from resource limitations.

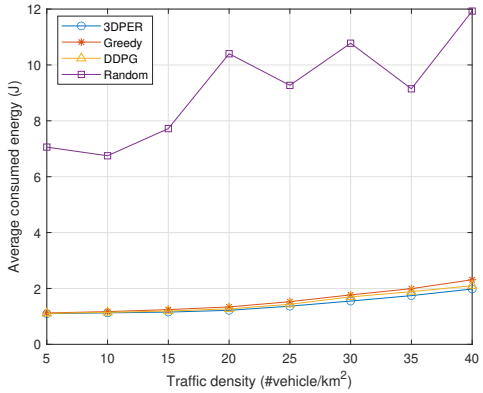

Fig. 14. Consumed energy under different traffic densities.

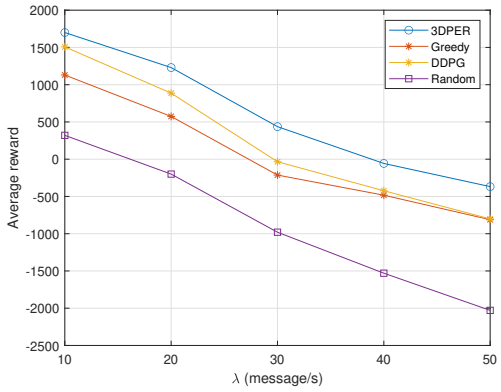

Fig. 15. Average reward under different message arrival rates.

Fig. 16 presents the results of the 3DPER algorithm's examination of the effect of integrated blockchain on BEVEC performance in terms of average energy consumption and latency across various delegate nodes. The results indicate that the average latency in the block verification process increases marginally as the number of delegate nodes increases. This is because the likelihood of selecting nodes at distant locations from each other increases with the number of delegate nodes, leading to slightly longer average latency. However, note that the number of delegate nodes itself does not have a significant impact on the block verification latency. Instead, the maximum of delays between delegates is the main factor in determining the overall block verification latency. In contrast, energy consumption is directly impacted by the number of delegate nodes. Each delegate node as a block verifier contributes to the total energy usage, resulting in a cumulative impact on the consumed energy. BEVEC performance is affected by the trade-off between increasing the number of delegate nodes and increasing the security of the blockchain. On one hand, this leads to a slight increase in average latency, but on the other hand, it significantly affects energy consumption.

VI. Conclusion and Future Work

In this paper, we proposed a secure framework for in-time delivery of V2X services by combining deep reinforcement learning and permissioned blockchain in vehicular edge computing networks. Our approach features a dual-layer verification process to ensure accurate and secure delivery of vehicular

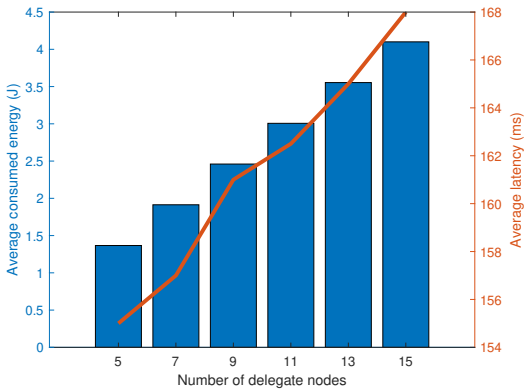

Fig. 16. Average consumed energy and latency under different delegate nodes.

messages. The first layer involves local verification to ensure the accuracy of disseminated messages, while the second layer uses a permissioned blockchain to guarantee the integrity of the messages. We introduced a new system utility that serves as a performance metric to ensure the prompt delivery of services as well as a measure for the selection of verifier nodes in the blockchain consensus mechanism. To optimize this utility function in the dynamic vehicular environment, we formulated the problem of in-time vehicular message delivery as a sequential decision problem and proposed a novel DRL-based algorithm - 3DPER - to solve it. Simulation results show that the proposed algorithm can effectively improve V2X service delivery performance in terms of energy consumption, latency, and success rate. However, further investigation into the limitations and potential enhancements of the proposed BEVEC framework is necessary. For instance, it would be worthwhile to investigate the suitability of BEVEC's architecture for services with strict delivery time requirements due to the dual layer verification. This could prove a limitation of the current architecture. In addition, although the 3DPER algorithm is currently capable of scaling with the number of vehicles, it is still a centralized approach, hence it is necessary to explore decentralized approaches, particularly in scenarios in which such scalability is critical. Our future efforts will focus on developing a decentralized algorithm that maintains security and trust while minimizing latency. Our objective is to expand the framework's utility across a wide range of application scenarios, to enhance its broad applicability.

REFERENCES

[1] V. Todisco, S. Bartoletti, C. Campolo, A. Molinaro, A. O. Berthet, and A. Bazzi, "Performance analysis of sidelink 5G-V2X mode 2 through an open-source simulator," *IEEE Access*, vol. 9, pp. 145648–145661, 2021.

[2] H. Masuda, O. E. Marai, M. Tsukada, T. Taleb, and H. Esaki, "Feature-based vehicle identification framework for optimization of collective perception messages in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 2120–2129, 2023.

[3] M. Fardad, G.-M. Muntean, and I. Tal, "Latency-aware V2X operation mode coordination in vehicular network slicing," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pp. 1–6, 2023.

[4] Y. Gong, Y. Wei, Z. Feng, F. R. Yu, and Y. Zhang, "Resource allocation for integrated sensing and communication in digital twin enabled internet of vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 4, pp. 4510–4524, 2023.

[5] Y. Qi, Y. Zhou, Y.-F. Liu, L. Liu, and Z. Pan, "Traffic-aware task offloading based on convergence of communication and sensing in vehicular edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17762–17777, 2021.

[6] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019.

[7] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, "Cooperative computation offloading in blockchain-based vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 783–798, 2022.

[8] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. Leung, "Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, pp. 1–15, 2023.

[9] S. S. Shinde, A. Bozorgchenani, D. Tarchi, and Q. Ni, "On the design of federated learning in latency and energy constrained computation offloading operations in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 71, pp. 2041–2057, Feb 2022.

[10] Y. Dai, D. Xu, K. Zhang, S. Maharjan, and Y. Zhang, "Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 4, pp. 4312–4324, 2020.

[11] A. Bozorgchenani, S. Maghsudi, D. Tarchi, and E. Hossain, "Computation offloading in heterogeneous vehicular edge networks: On-line and off-policy bandit solutions," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4233–4248, 2022.

[12] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[13] M. Hasan, S. Mohan, T. Shimizu, and H. Lu, "Securing vehicle-to-everything (V2X) communication platforms," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 693–713, 2020.

[14] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, "Integrated blockchain and edge computing systems: A survey, some research issues and challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1508–1532, 2019.

[15] J. Kang, M. Qiu, J. Ma, and C.-X. Wang, "Blockchain for secure and efficient data sharing in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, pp. 4660–4670, Jun 2019.

[16] Y. Lin, J. Kang, D. Niyato, Z. Gao, and Q. Wang, "Efficient consensus and elastic resource allocation empowered blockchain for vehicular networks," *IEEE Transactions on Vehicular Technology*, pp. 1–6, 2022.

[17] S. Ma, S. Wang, and W.-T. Tsai, "Delay analysis of consensus communication for blockchain-based applications using network calculus," *IEEE Wireless Communications Letters*, vol. 11, no. 9, pp. 1825–1829, 2022.

[18] S. Wang, D. Ye, X. Huang, R. Yu, Y. Wang, and Y. Zhang, "Consortium blockchain for secure resource sharing in vehicular edge computing: A contract-based approach," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1189–1201, 2021.

[19] P. Liu, W. Jing, X. Fu, Y. Xiao, and L. Jia, "Consortium blockchain-based security and efficient resource trading in V2V-assisted intelligent transport systems," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2023.

[20] J. Gao, Z. Kuang, J. Gao, and L. Zhao, "Joint offloading scheduling and resource allocation in vehicular edge computing: A two layer solution," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 3, pp. 3999–4009, 2023.

[21] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 7944–7956, 2019.

[22] Q. Luo, C. Li, T. H. Luan, W. Shi, and W. Wu, "Self-learning based computation offloading for internet of vehicles: Model and algorithm," *IEEE Transactions on Wireless Communications*, vol. 20, no. 9, pp. 5913–5925, 2021.

[23] B. Shang, L. Liu, and Z. Tian, "Deep learning-assisted energy-efficient task offloading in vehicular edge computing systems," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9619–9624, 2021.

[24] Z. Ning, P. Dong, X. Wang, X. Hu, J. Liu, L. Guo, B. Hu, R. Y. K. Kwok, and V. C. M. Leung, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 21, no. 4, pp. 1319–1333, 2022.

[25] P. Li, Z. Xiao, X. Wang, K. Huang, Y. Huang, and H. Gao, "EPtask: Deep reinforcement learning based energy-efficient and priority-aware

task scheduling for dynamic vehicular edge computing," *IEEE Transactions on Intelligent Vehicles*, pp. 1–17, 2023.
[26] Q. Fan, L. Chen, C. You, Y. Chen, and H. Yin, "Dependency-aware service migration for backhaul-free vehicular edge computing networks," *IEEE Transactions on Vehicular Technology*, pp. 1–16, 2023.
[27] V. Huy Hoang, T. M. Ho, and L. B. Le, "Mobility-aware computation offloading in MEC-based vehicular wireless networks," *IEEE Communications Letters*, vol. 24, no. 2, pp. 466–469, 2020.
[28] W. Zhan, C. Luo, J. Wang, C. Wang, G. Min, H. Duan, and Q. Zhu, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things Journal*, vol. 7, no. 6, pp. 5449–5465, 2020.
[29] C. Li, F. Liu, B. Wang, C. L. P. Chen, X. Tang, J. Jiang, and J. Liu, "Dependency-aware vehicular task scheduling policy for tracking service VEC networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2400–2414, 2023.
[30] P. Dai, K. Hu, X. Wu, H. Xing, F. Teng, and Z. Yu, "A probabilistic approach for cooperative computation offloading in MEC-assisted vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 899–911, 2022.
[31] B. Ma, Z. Ren, and W. Cheng, "Traffic routing-based computation offloading in cybertwin-driven internet of vehicles for V2X applications," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4551–4560, 2022.
[32] H. Wang, T. Lv, Z. Lin, and J. Zeng, "Energy-delay minimization of task migration based on game theory in MEC-assisted vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8175–8188, 2022.
[33] E. M. Mianji, G.-M. Muntean, and I. Tal, "Trustworthy routing in VANET: A Q-learning approach to protect against black hole and gray hole attacks," in *2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring)*, pp. 1–6, 2023.
[34] X. Wang, H. Zhu, Z. Ning, L. Guo, and Y. Zhang, "Blockchain intelligence for internet of vehicles: Challenges and solutions," *IEEE Communications Surveys Tutorials*, vol. 25, no. 4, pp. 2325–2355, 2023.
[35] G. S. Aujla, A. Singh, M. Singh, S. Sharma, N. Kumar, and K.-K. R. Choo, "BloCkEd: Blockchain-based secure data processing framework in edge envisioned V2X environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5850–5863, 2020.
[36] D. Zhang, F. R. Yu, and R. Yang, "Blockchain-based multi-access edge computing for future vehicular networks: A deep compressed neural network approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12161–12175, 2022.
[37] L. Cui, Z. Chen, S. Yang, Z. Ming, Q. Li, Y. Zhou, S. Chen, and Q. Lu, "A blockchain-based containerized edge computing platform for the internet of vehicles," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2395–2408, 2021.
[38] X. Zheng, M. Li, Y. Chen, J. Guo, M. Alam, and W. Hu, "Blockchain-based secure computation offloading in vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4073–4087, 2021.
[39] Y. Ren, X. Chen, S. Guo, S. Guo, and A. Xiong, "Blockchain-based VEC network trust management: A DRL algorithm for vehicular service offloading and migration," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 8, pp. 8148–8160, 2021.
[40] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3882–3897, 2023.
[41] F. Jameel, M. A. Javed, S. Zeadally, and R. Jäntti, "Efficient mining cluster selection for blockchain-based cellular V2X communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4064–4072, 2021.
[42] M. Alsenwi, N. H. Tran, M. Bennis, S. R. Pandey, A. K. Bairagi, and C. S. Hong, "Intelligent resource slicing for eMBB and URLLC coexistence in 5G and beyond: A deep reinforcement learning based approach," *IEEE Transactions on Wireless Communications*, vol. 20, no. 7, pp. 4585–4600, 2021.
[43] C. Chaieb, F. Abdelkefi, and W. Ajib, "Deep reinforcement learning for resource allocation in multi-band and hybrid OMA-NOMA wireless networks," *IEEE Transactions on Communications*, vol. 71, no. 1, pp. 187–198, 2023.
[44] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013.
[45] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*, pp. 1995–2003, PMLR, 2016.
[46] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.
[47] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.
[48] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: The MIT Press, 2nd ed., 2018.
[49] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 3053–3062, PMLR, 10–15 Jul 2018.
[50] M. Parvini, M. R. Javan, N. Mokari, B. Abbasi, and E. A. Jorswieck, "Aoi-aware resource allocation for platoon-based C-V2X networks via multi-agent multi-task reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 9880–9896, 2023.

Mohammad Fardad received the M.S. degree in electrical engineering from Shahid Beheshti University, Tehran, Iran in 2014. He is currently pursuing a PhD degree from the School of Computing, Dublin City University, Dublin, Ireland. His research interests include blockchain, mobile edge computing and deep reinforcement learning with emphasis on vehicular networks. He is a recipient of an SFI scholarship from the Lero Centre for Software at Dublin City University, Ireland.

Gabriel-Miro Muntean (Fellow, IEEE) received his Ph.D. degree for research on adaptive multimedia delivery from Dublin City University (DCU), Ireland, in 2004, where he is a Professor with the School of Electronic Engineering and co-Director of the DCU Performance Engineering Laboratory. He has published over 500 papers in top-level international journals and conferences, authored four books and 29 book chapters, and edited nine additional books and proceedings. His research interests include quality, performance, and energy saving issues related to multimedia and multiple sensorial media delivery, technology-enhanced learning, and other data communications over heterogeneous networks. He is an Associate Editor of the IEEE TRANSACTIONS ON BROADCASTING, the Multimedia Communications Area Editor of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and the chair and a reviewer for important international journals, conferences, and funding agencies. He is a Fellow of IEEE Broadcast Technology Society.

Irina Tal (Senior Member, IEEE) Irina Tal received the Ph.D. degree from the School of Electronic Engineering, Dublin City University, Ireland, where she is an Assistant Professor with the School of Computing, an Academic Lead of the M.Sc. in Blockchain, and a member of LERO. She is the Lead Principal Investigator on the SFI funded project PRIVATT. She published in prestigious international conferences and journals. Her research interests include technology-enhanced learning, vehicular ad-hoc networks, smart cities, and cyber security.