

Article

Two-Layer Blockchain-Based Secure Task Offloading of Vehicle Edge Computing

Guoling Liang ^{1,2} , Feng Zhao ^{1,*} , Chunhai Li ¹ , Zhaoyu Su ¹  and Mingfeng Chen ¹

¹ Guangxi Engineering Research Center of Industrial Internet Security and Blockchain, Guilin University of Electronic Technology, Guilin 541004, China; guoliang@mails.guet.edu.cn (G.L.); chunhaili@guet.edu.cn (C.L.); szyguet@guet.edu.cn (Z.S.); cmf@guet.edu.cn (M.C.)

² School of Physics and Telecommunication Engineering, Yulin Normal University, Yulin 537000, China

* Correspondence: zhaofeng@guet.edu.cn

Abstract

Blockchain provides a secure and trusted environment for task offloading in Vehicle Edge Computing (VEC). Nevertheless, single-layer blockchain architecture exhibits low transaction processing efficiency and insufficient scalability in VEC environments. Accordingly, we present a two-layer blockchain-based VEC (TBVEC) task offloading framework, which extends consensus nodes to parked vehicles (PVs) to improve the scalability of edge computing. Specifically, a Hotstuff-BLSPBFT two-layer blockchain is constructed by employing Hotstuff for the lower-layer blockchain and BLSPBFT for the upper-layer blockchain, thereby improving transaction efficiency of the blockchain system. Then, sharding is performed on consensus nodes of the lower-layer blockchain, and an improved K-Means algorithm is adopted for grouping optimization. Furthermore, utility functions of user vehicles and Road Side Units (RSUs) are formulated based on a two-layer blockchain. A game-based Deep Reinforcement Learning (DRL) model, equipped with two Soft Actor-Critic (SAC) agents, is developed to optimize user and RSU utility functions. The model determines the optimal task offloading strategies and transaction pricing policies, while jointly optimizing the number of consensus node groups and the quantity of nodes within each group to maximize the utility function. Experimental results validate that the proposed scheme not only ensures the security of task offloading but also significantly enhances overall system efficiency.

Keywords: task offloading; blockchain; VEC; Hotstuff; BLSPBFT

MSC: 68M10



Academic Editors: Fasheng Zhou and Jing Qiu

Received: 5 June 2026

Revised: 3 July 2026

Accepted: 10 July 2026

Published: 12 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

With the rapid development of wireless communication technologies and artificial intelligence, the Internet of Vehicles (IoV) has entered a new stage of intelligence. However, restricted by multiple constraints such as in-vehicle space, manufacturing cost, and energy supply, the computing and storage resources of on-board terminals are relatively limited, making it difficult to meet the requirements of large-scale data processing and the training of complex deep learning models. Cloud computing has been regarded as a promising solution to overcome the above limitations by providing centralized resources to accomplish computation-intensive tasks [1,2]. Nevertheless, cloud servers are usually deployed far away from end users, resulting in inherent drawbacks of cloud computing, including

high network latency and limited bandwidth, which make it unsuitable for safety-critical applications requiring ultra-low latency [3].

Mobile Edge Computing (MEC) is an emerging computing paradigm that directly addresses this challenge [4,5]. Compared with traditional cloud computing, MEC shortens the physical distance between end devices and servers, thereby effectively reducing data transmission latency [6,7]. By deploying servers at the edge of network, computation tasks generated by user applications can be processed directly at the edge side, not only alleviating the pressure on the core network but also optimizing user experience. With the deep integration of MEC and IoV, VEC has emerged as a key paradigm [8,9]. By offloading computation tasks to edge nodes such as RSUs, VEC effectively reduces task processing latency and improves system efficiency. However, task offloading in VEC faces numerous security challenges. For instance, when transaction data of vehicular edge nodes is stored in distributed databases, it is critical to safeguard sensitive and private information of these nodes against leakage and abuse [10]. Furthermore, trust issues among network nodes in IoV composed of RSUs and parked vehicles have become increasingly critical for data security and network stability, demanding effective solutions [11]. To address the above issues, researchers have integrated blockchain with task offloading, leveraging its decentralization, tamper-proof nature, and traceability to manage transactions during the offloading process, thereby constructing a secure and trustworthy offloading environment [12]. However, there still remain several challenges to be resolved when combining blockchain with VEC task offloading. For instance, cross-domain task offloading transactions suffer from low transaction efficiency and insufficient scalability. Furthermore, how to balance the utilities of users and edge servers in a dynamic environment while ensuring security also presents a nontrivial challenge.

To address the above issues, we propose TBVEC to improve transaction efficiency and scalability and enable PVs to participate in blockchain consensus to improve system scalability. The offloading strategy, pricing strategies of RSUs, as well as the number of groups and intra-group nodes in the lower-layer blockchain are optimized, so as to ensure system revenue while achieving performance balance of the consensus mechanism in each group. The major contributions of this work are listed below.

1. We propose a TBVEC task offloading framework, which extends consensus nodes to PVs and adopts Hotstuff as the consensus mechanism for the lower-layer blockchain and BLSPBFT for the upper-layer blockchain, thereby improving transaction efficiency of the task offloading process.
2. According to the characteristics of the consensus mechanism in the lower-layer blockchain, we present an improved Hotstuff consensus scheme based on K-Means algorithm. The consensus nodes in the lower layer are grouped with respect to their energy consumption and consensus latency, aiming to achieve performance balance across different groups.
3. We establish utility functions of users and RSUs, incorporate the energy consumption and latency caused by the consensus mechanism into the utility functions, and formulate a joint optimization problem. A game-based DRL model with two SAC agents is constructed to optimize the utility functions and maximize their respective utilities.
4. We conduct extensive simulations to validate the performance of the proposed mechanism. Results demonstrate that our solution can notably enhance system scalability while protecting the benefits of the system.

The remainder of this paper is organized as follows. Section 2 reviews the recent related work on blockchain-based task offloading and DRL task offloading. Section 3 elaborates the system model of TBVEC task offloading. Section 4 analyzes the utility functions of users and RSUs and establishes the joint optimization problem. Section 5

presents the improved Hotstuff consensus algorithm based on K-Means algorithm and establishes the game-based DRL model for the optimization problem. Section 6 evaluates the proposed scheme by experiments. Finally, Section 7 concludes this paper.

2. Related Work

2.1. Blockchain for Secure Task Offloading

Traditional VEC relies on edge servers or cloud servers to centrally manage vehicle information and transactions during task offloading, making the system prone to a single point of failure and vulnerable to data tampering. To address this issue, researchers have integrated blockchain technology into edge computing. For instance, the literature [13] deployed blockchain on the RSU network to manage transactions during the offloading process, thereby constructing a distributed and trustworthy offloading environment. And the works in [14,15] utilized smart contracts to manage request publication, task execution, and incentive distribution. However, these studies have neither considered the impact of blockchain on the system nor addressed how to improve transaction efficiency and scalability. The work presented in [16] utilized Fair Proof of Reputation (FPoR) to support secure and decentralized task offloading, with guaranteed data integrity and reliability. The work in [17] utilized a smart contract-driven Proof of Stake (PoS) consensus mechanism to implement dynamic pricing for task offloading transactions, realizing secure, scalable, and efficient computation resource management. Nonetheless, the above works adopt public blockchains, in which every distributed node is obligated to participate in the consensus process, resulting in long block generation and validation latency as well as high energy consumption, which are incompatible with energy-constrained and delay-sensitive vehicular networks [18]. To address these issues, some works adopt consortium or even private blockchains deployed on edge servers. In [19], consensus nodes were directly deployed on RSUs, and a secure scheme combining the Raft consensus protocol with edge computing was developed, which alleviated the additional latency introduced by blockchain. For delay-sensitive applications, fully exploiting the communication capacity of consensus nodes to reduce the time and frequency of block propagation represented another effective solution [20]. However, the above studies overlooked the balance between consensus performance and system performance. Therefore, Lang et al. [21] presented a collaborative computation offloading (CO) technique and a secure handover framework with consensus mechanisms, and designed a collaborative CO decision optimization scheme combined with PBFT to achieve secure switching, ensuring the efficiency and security of collaborative CO and secure handover. In [22], the authors introduced an innovative proof-of-reputation consensus scheme underpinned by a lightweight block verification mechanism. By co-optimizing offloading decisions, channel selection, transmission power allocation, and computation resource allocation, this work achieved the maximization of system utility for blockchain-based MEC systems. Moreover, enhancing consensus mechanisms and performing joint optimization with system factors can significantly promote the overall system performance. For instance, Zhang et al. [23] proposed the RPBFT consensus algorithm, which balances communication resource utilization, computational resource utilization, and transmission delay, thereby improving the accuracy, timeliness, and security of resource allocation. Some works in [24–26] selected nodes with high reliability, favorable communication quality, or abundant computational resources as consensus nodes to refine consensus mechanisms and optimize task offloading, which greatly improved system performance. However, traditional consortium or private blockchain consensus mechanisms still suffer from scalability issues in task offloading. Specifically, with the growth in the number of nodes, consensus mechanisms are faced with high communication overhead, which leads to substantial performance degradation. To some extent, cross-chain

technologies offer a viable solution to this problem, as reported in [27,28]. In addition, sharding technology is also an effective method to address the scalability issue. Therefore, to solve the inherent problem of low time efficiency of blockchain, the work in [29] proposed a blockchain sharding scheme to reduce the system time latency, and similar work has also been done in [30,31]. However, the above works neglect the performance balance of consensus mechanisms across different groups when adopting cross-chain and sharding technologies, and also fail to consider system performance for task offloading. Consequently, further improvements in transaction processing efficiency and scalability are hindered, making it difficult to maximize system utility.

2.2. DRL for Task Offloading

The dynamic VEC environment makes it difficult for traditional optimization methods to optimize long-term cumulative rewards and make dynamic decisions under low-latency constraints. Therefore, many studies have begun to apply DRL algorithms to dynamic offloading decisions. In [32], the authors employed TD3 to optimize the uplink transmit power allocation sub-problem, thereby minimizing the overall task processing cost of all devices. Meanwhile, the study in [33] utilized MADRL, with RSUs acting as agents to participate in decision-making, to optimize network resources and maximize the average number of offloading tasks. And the works in [26,34] conducted similar studies by using SAC and DQN to make dynamic decisions on offloading and resource allocation, respectively, thereby minimizing the total system delay and energy consumption. However, none of the above works considered the applicability of directly applying existing single DRL algorithms to VEC, thereby failing to further enhance system performance. Therefore, the literature [35] formulated a joint blockchain, communication, computation, and caching (B3C) optimization problem and employed the multi-agent double actor-critic (MADAC) algorithm to reduce task processing latency and improve blockchain transaction throughput. The literature [36] adopted an improved DDGP to make decisions on continuous variables such as vehicle distance, task size, and vehicle speed according to the dynamic changes in vehicle mobility and channel conditions, thereby achieving efficient resource allocation. Similar improvement was also investigated in [37]. However, these studies merely refined existing DRL algorithms without overcoming the limitations of single-agent or homogeneous multi-agent architectures, thereby failing to accommodate the intricate coupling and competition among multiple vehicles and RSUs. Therefore, the literature [38] adopted a two-stage DRL algorithm and improved DDQN to avoid value overestimation and low exploration efficiency, thereby achieving collaborative content caching decisions. And the work in [39] proposed a dual-layer architecture based on TD3 and D3QN to minimize system latency, ensuring both offloading security and low latency while adapting to the highly dynamic VEC network environment. The studies in [40,41] combined traditional optimization with DRL to overcome the limitations of single-agent or homogeneous multi-agent architectures, forming a hybrid architecture that improved system performance. However, due to the competition between users and edge servers in VEC, these architectural improvements failed to balance the utilities of all parties, reducing the overall system utility.

3. System Model

3.1. TBVEC Offloading System Framework

The framework and workflow of task offloading for TBVEC are illustrated in Figure 1. The system architecture mainly comprises users, PVs, RSUs and VEC servers. The PVs and RSUs form a two-layer blockchain network, where the lower-layer blockchain consists of PVs and RSUs, and the number of PVs exceeds that of RSUs. Each shard can execute the

Hotstuff consensus algorithm in parallel. The upper-layer blockchain is formed by the RSU network, which implements the BLSPBFT consensus algorithm.

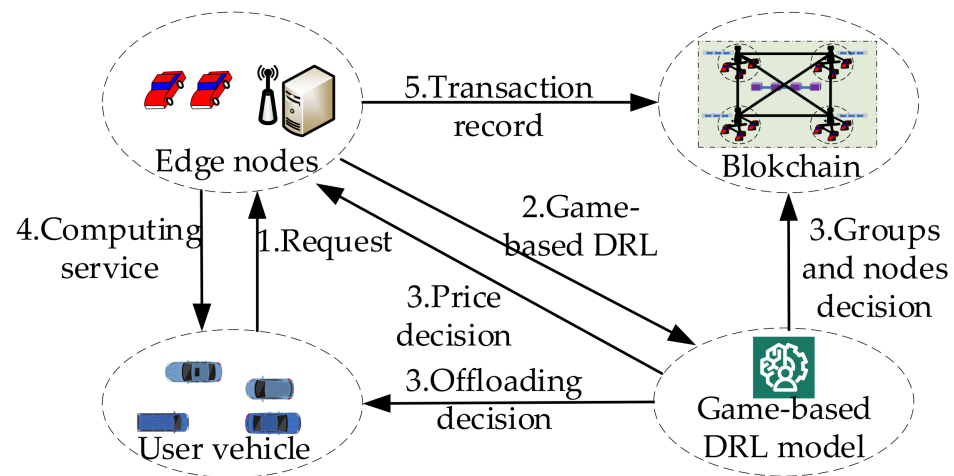


Figure 1. TBVEC offloading system framework and workflow.

When a user requires computation services, it sends a service request to the edge node. Upon verifying the legitimacy of the user, the edge node allows the user to offload its computational tasks. The edge node then executes the game-based DRL model deployed on the edge server, so as to generate optimal decisions regarding task offloading schemes, pricing policies for RSUs, the number of consensus node groups, and the number of nodes within each group. Based on these optimized decision variables, the edge node provides computing services for the user and generates a corresponding service transaction. The transaction record is subsequently uploaded to the blockchain for storage and verification. The key notations used in this paper are summarized in Table 1.

Table 1. Explanation of notations.

Notations	Explanation
$C_{vi}^{exe}, C_{rj}^{exe}$	CPU cycles per unit task size required for RSU and vehicle computing
d_0	Wireless far-field reference distance
$d_{i,j}, d_{i,k}$	Distance from RSU or a vehicle to the receiving vehicle and the interfering vehicle
D_{qi}	Task size for user i
D_{Bv}	Block size
E_{RSUi}	RSU energy consumption for computational tasks
E_{useri}	User energy consumption for offloading tasks
f_{vk}, f_{rj}	Computational capacity of vehicle k and RSU j
N_0	Noise power of the wireless channel
P_t	Wireless transmission power
$R_{i,j}$	Communication transmission rate
T_{useri}, T_{RSUi}	User and RSU delay for processing computational tasks
W_b	Wireless communication bandwidth
α	Path loss factor
β	CPU cycles required for generating or verifying signatures
ε_{RSUi}	Proportion of task offloading to RSU by user i
η	Transceiver decision factor
θ	CPU cycles required for generating or verifying the Message Authentication Code (MAC)
κ_v, κ_r	Capacitive switching coefficients for vehicles and RSUs
ω	Average transaction size of blockchain

3.2. Network Model

In this work, we adopt a time slot model to partition the total time T into discrete time slots. In each time slot t , the communication links among vehicles and between vehicles and edge nodes are assumed to be static. Assume that there are U users and R RSUs in the considered area, with their sets denoted as $U = \{1, 2, \dots, U\}$ and $R = \{1, 2, \dots, R\}$, respectively. According to references [42,43], the SINR of RSU or a vehicle receiving from other vehicles is expressed as:

$$\text{SINR}_i(t) = \frac{P_t \eta (d_0/d_i(t))^\alpha}{N_0 + \sum_{j=1, j \neq i}^N [P_t \eta (d_0/d_{i,j}(t))^\alpha]} \quad (1)$$

where P_t denotes the transmission power, η is the transceiver determination coefficient, d_0 and $d_i(t)$ are the far-field reference distance and distance from RSU or a vehicle to the receiving vehicle, respectively, $d_{i,j}(t)$ is the distance from the RSU or a vehicle to the interfering vehicle, N_0 stands for the noise power, and α denotes the path loss exponent. Accordingly, the wireless communication transmission rate between nodes can be given as

$$R_{i,j}(t) = W_B \times \log_2 \left(1 + \frac{P_t h(d_0/d_{i,j}(t))^a}{N_0 + \sum_{k=1, k \neq i, k \neq j}^N [P_t h(d_0/d_{i,k}(t))^a]} \right) \quad (2)$$

where W_B denotes the wireless communication bandwidth, $d_{i,j}(t)$ and $d_{i,k}(t)$ represent the distance from a vehicle or RSU to the receiving vehicle and from a vehicle or RSU to the interfering vehicle, respectively.

3.3. Task Offloading Model

Within time slot t , let $\varepsilon_{RSUi}(t)$ denote the proportion of tasks offloaded by user i to the RSU. Then, the delays of local computing and incurred by computation services offloaded to the RSU are expressed as

$$T_{useri}(t) = \frac{(1 - \varepsilon_{RSUi}(t)) D_{qi} C_{vi}^{exe}}{f_{vi}(t)} \quad (3)$$

$$T_{RSUi}(t) = \max_j \left(\frac{\varepsilon_{RSUi}(t) D_{qi} C_{rj}^{exe}}{f_{rj}(t)} + \frac{\varepsilon_{RSUi}(t) D_{qi}}{R_{i,j}(t)} \right) \quad (4)$$

where $i \in U$, $j \in R$, $f_{rj}(t)$ and $f_{vi}(t)$ are the computing capacities of RSU j and vehicle i , respectively, D_{qi} is the total task size of user i , and C_{vi}^{exe} , C_{rj}^{exe} denote the required CPU cycles for processing a unit task size at vehicle i and RSU j , respectively.

Furthermore, during the process of task offloading, the energy consumption incurred by the users and RSUs can be respectively expressed as

$$E_{useri}(t) = \left(P_t \frac{\varepsilon_{RSUi}(t) D_{qi}}{R_{i,j}(t)} + \kappa_v f_{vi}(t)^2 (1 - \varepsilon_{RSUi}(t)) D_{qi} C_{vi}^{exe} \right) \quad (5)$$

$$E_{RSUi}(t) = \sum_{j=1}^R \left(\kappa_r f_{rj}(t)^2 \varepsilon_{RSUi}(t) D_{qi} C_{rj}^{exe} \right) \quad (6)$$

In the above equations, κ_v and κ_r are the capacitance switching coefficients of vehicles and RSUs, respectively, and P_t denotes the wireless transmission power.

3.4. Two-Layer Blockchain Model

In this paper, Hotstuff is employed as the consensus mechanism for the lower-layer blockchain, and BLS-PBFT for the upper-layer blockchain. The two-layer consensus procedure is illustrated in Figure 2. The overall consensus process of the system can be

divided into two phases. First, the consensus nodes in the lower layer are grouped using the improved K-Means algorithm, followed by the execution of the Hotstuff consensus mechanism. Then, the leader nodes of each group in the lower layer form the upper-layer consensus nodes to execute BLSPBFT. Notably, the leader node of each lower-layer group is selected from RSUs. Assume that during the consensus process, the numbers of CPU cycles required for generating or verifying signatures and MACs are β and θ , respectively. The lower-layer blockchain is divided into $S(t)$ groups, each containing $L(t)$ consensus nodes, among which M RSUs and K PVs participate in the consensus, satisfying $L(t) = M + K$. Let F denote the number of Byzantine nodes, and then the number of consensus nodes in the upper-layer blockchain is $S(t)$. Let the primary node be B_l and replica nodes be B_m , where $B_l, B_m \in \{1, 2, 3, \dots, L\}$. Let $D_{Bv}(t)$ and ω denote the block size and the average transaction size, respectively. According to the work in [44], the computational energy consumption incurred by PVs and RSUs during the execution of Hotstuff consensus in the lower-layer blockchain is given by:

$$E_{pa}^{Hc}(t) = \sum_{m=1}^K \left(\kappa_v f_{pm}(t)^2 \left(\left(\frac{D_{Bv}(t)}{\omega} + 7 \right) \beta + \left(\frac{D_{Bv}(t)}{\omega} + 7 \right) \theta \right) \right) \quad (7)$$

$$E_{RSU}^{Hc}(t) = \left(\frac{D_{Bv}(t)}{\omega} + 8 \right) \beta + \left(\frac{D_{Bv}(t)}{\omega} + 8F + 5 \right) \theta + \sum_{m=1}^M \left(\kappa_r f_{rm}(t)^2 \left(\left(\frac{D_{Bv}(t)}{\omega} + 7 \right) \beta + \left(\frac{D_{Bv}(t)}{\omega} + 7 \right) \theta \right) \right) \quad (8)$$

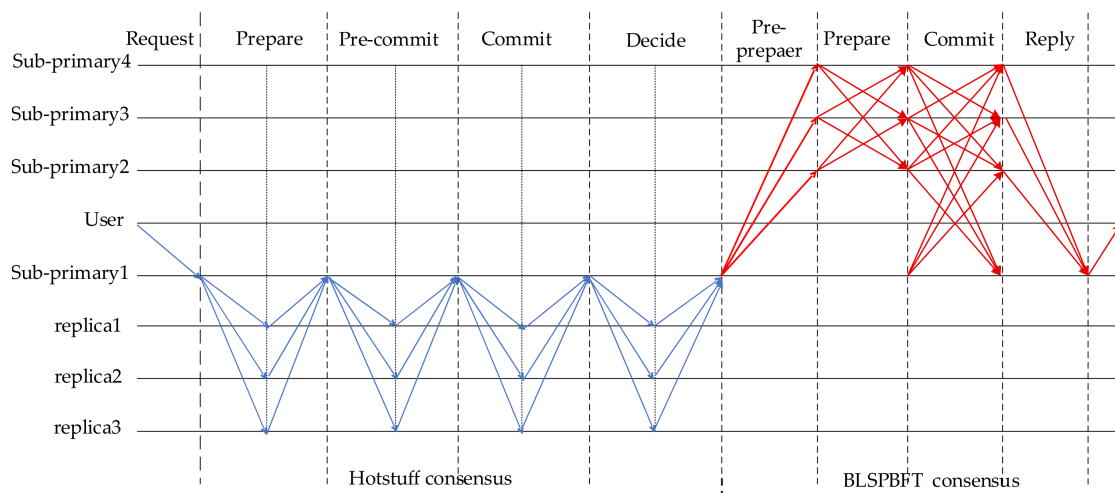


Figure 2. The consensus process of two-layer blockchain.

Let F' denote the number of Byzantine nodes in the upper-layer blockchain. Then, according to [45], the computational energy consumption incurred by RSUs when executing the BLSPBFT consensus in the upper-layer blockchain can be expressed as:

$$E_{RSU}^{Pc}(t) = \left(\frac{D_{Bv}(t)}{\omega} + 8 \right) \beta + \left(\frac{D_{Bv}(t)}{\omega} + 6F' + 2L(t) - 2 \right) \theta + \sum_{m=1}^M \left(\kappa_r f_{rm}(t)^2 \left(\left(\frac{2D_{Bv}(t)}{\omega} + 7 \right) \beta + \left(\frac{2D_{Bv}(t)}{\omega} + 4F' + 2L(t) - 1 \right) \theta \right) \right) \quad (9)$$

As for the communication energy consumption incurred by consensus execution, the communication pattern of Hotstuff in the lower layer is that either the primary node sends messages to all replica nodes, or all replica nodes send messages to the primary node. In addition, communications among RSUs are wired, while communications among other

nodes are wireless. Let R_w be the wired communication rate; the communication energy consumption of PVs and RSUs during the consensus execution is given by

$$E_{pa}^{HT}(t) = 4 \times \left(\sum_{m=1}^K \frac{D_{Bv}(t)}{R_{Bm,Bl}(t)} \right) \times P_t \quad (10)$$

$$E_{RSU}^{HT}(t) = 4 \times \left(\sum_{m=1}^{M-1} \frac{D_{Bv}(t)}{R_w} + \sum_{m=1}^K \frac{D_{Bv}(t)}{R_{Bl,Bm}(t)} \right) \times P_t \quad (11)$$

During the execution of BLSPBFT in the upper layer, the pre-prepare phase involves the primary node sending messages to replica nodes, the prepare and commit phases involve mutual message exchange among nodes, and the reply phase consists of replica nodes sending messages to the primary node. Since the entire process adopts wired communication, the communication energy consumption incurred by RSUs is expressed as

$$E_{RSU}^{PT}(t) = P_t \times \frac{D_{Bv}(t)}{R_w} [2(M-1) + (M-2)(M-1) + M(M-1)] \quad (12)$$

According to the above analysis, the overall energy consumption of PVs and RSUs throughout the two-layer blockchain consensus is given by:

$$E_{pa}^B(t) = \frac{U}{D_{Bv}(t)/\omega} S(t) \times (E_{pa}^{Hc}(t) + E_{pa}^{HT}(t)) \quad (13)$$

$$E_{RSU}^B(t) = \frac{U}{D_{Bv}(t)/\omega} \left(S(t) \times (E_{RSU}^{Hc}(t) + E_{RSU}^{HT}(t)) + E_{RSU}^{Pc}(t) + E_{RSU}^{PT}(t) \right) \quad (14)$$

4. Problem Format

4.1. User Offloading Allocation Problem

The user's utility function is quantified as the difference between its computation time satisfaction and the total cost of task execution. Let p_{RSUi} denote the unit task size price provided by RSUs. Then the utility function can be expressed as

$$U_{useri}(t) = \lambda(T_{maxi} - T_i(t))^2 - \varepsilon_{RSUi} D_{qi} p_{RSUi}(t) - \xi_v E_{useri}(t) \quad (15)$$

where λ is the satisfaction coefficient, ξ_v denotes the energy consumption cost for processing a unit task size at the vehicle, T_{maxi} is the maximum tolerable computation service delay for user i , and $T_i = \max(T_{useri}, T_{RSUi})$. In Equation (15), the first term depicts the user's satisfaction with the computation delay, the second represents the computation service cost of RSUs, and the last term is the energy consumption cost of the user. The user intends to complete task computation with the shortest delay and the lowest cost, i.e., to maximize the utility function, which can be expressed as follows:

P1:

$$\max_{\varepsilon_{RSUi}(t)} U_{useri}(t) \quad (16)$$

$$\text{s.t. } T_{useri}(t), T_{RSUi}(t) \leq T_{maxi} \quad (17)$$

$$0 \leq \varepsilon_{RSUi}(t) \leq 1 \quad (18)$$

4.2. RSU Pricing Problem

RSU nodes generate revenue by providing computing services for users, but incur energy consumption costs during the computation process and consensus. Therefore, the utility function of the RSU is given by

$$U_{RSUi}(t) = \varepsilon_{RSUi}(t)D_{qi}p_{RSUi}(t) - \zeta_r \left[E_{RSUi}(t) + E_{RSU}^B(t)/U \right] - \zeta_v E_{pa}^B(t)/U \quad (19)$$

where ζ_r is the energy consumption cost of the RSU for processing unit task size. The first term represents the revenue of the RSU obtained through bargaining with users, and the second term is the energy consumption cost incurred by the RSU in task execution and consensus mechanism. The RSU maximizes its utility function to achieve revenue maximization.

P2:

$$\max_{p_{RSUi}(t)} U_{RSUi}(t) \quad (20)$$

$$\text{s.t. } U_{RSUi}(t) \geq 0 \quad (21)$$

5. TBVEC Task Offloading Strategy

5.1. K-Means-Based Two-Layer Blockchain Consensus Algorithm

In the two-layer blockchain model, the consensus nodes of the lower-layer blockchain consist of parking nodes and RSU nodes, and grouping optimization is performed on these lower-layer consensus nodes. Directly applying the traditional K-Means algorithm to blockchain consensus grouping in VEC scenarios has the following limitations:

- (1) Since the traditional K-Means algorithm relies on physical coordinates for clustering, whereas consensus efficiency is mainly governed by nodes' computing power and communication quality, nodes in close proximity may have substantially divergent performance. Hence, grouping by location may degrade consensus efficiency.
- (2) Traditional K-Means uses random centroid initialization and cannot guarantee at least one RSU per cluster; clusters without RSUs will have substantially different performance from those with RSUs, undermining their transaction processing efficiency.
- (3) Traditional K-Means demands a preset number of clusters K , which cannot be fixed in advance due to the dynamic nature of VEC environments, where node scale and task load vary over time.

To overcome the shortcomings of traditional K-Means, this paper improves the K-Means algorithm by taking the computing capacity and communication quality of nodes as clustering criteria. DRL is adopted to make optimal decisions regarding the number of groups and the number of nodes within each group. Correspondingly, an enhanced K-Means algorithm is proposed to balance node performance. Furthermore, combined with the upper-layer blockchain, the latency and energy consumption of PVs and RSUs are derived, the details of which are presented in Algorithm 1.

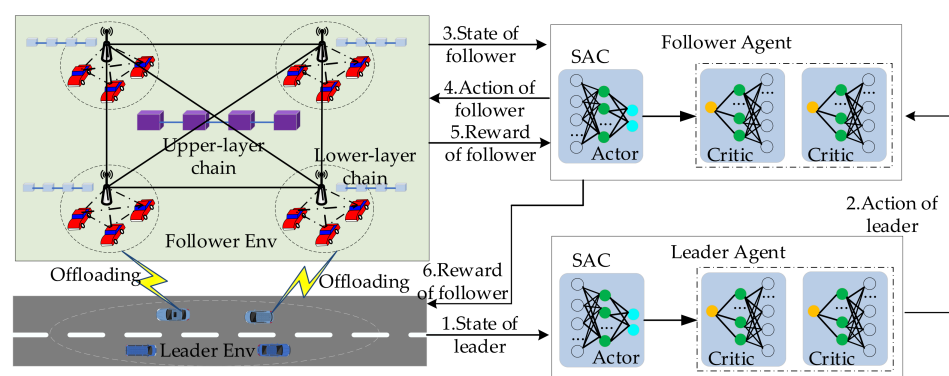
Algorithm 1 first sorts nodes by priority according to computing capacity and SNR, where RSUs have the highest priority. It selects $S \times L$ consensus nodes and chooses S RSUs as cluster centers for grouping. Then, it scores the selected nodes in each group based on computing capacity and SNR. If the score fluctuation exceeds a certain threshold, nodes are adjusted to balance the scores across groups. Finally, the consensus nodes of each group are determined, and the consensus delay and energy consumption of each group are calculated. Together with the upper-layer blockchain, the delay and energy consumption of PVs and RSUs during the consensus process are obtained, respectively. The upper-layer blockchain in this paper adopts BLSPBFT and the lower-layer employs Hotstuff. According to the communication complexity of BLSPBFT and Hotstuff, the communication complexity of Algorithm 1 is $O(L + S^2)$.

Algorithm 1: K-Means-based Two-layer Blockchain Consensus Algorithm**Input:** PV neodes: pa ; RSU nodes: RSU ;Group number S ; intra-group consensus nodes L **Output:** PV and RSU consensus energy consumption E_{pa}^B , E_{RSU}^B

- 1: Select long stay time PV from pa : $Exepa$
- 2: $AllNodes \leftarrow RSU \cup Exepa$
- 3: $isRSU \leftarrow [true: RSU, false: Exepa]$
- 4: Compute SNR of $Exepa$ by position (x, y)
Compute priority $\leftarrow$
- 5: $RSUpriority \times isRSU + a_1 \left(f_{pi} / \sum_{Exepa} f_{pi} \right) + a_2 \left(SNR_i / \sum_{Exepa} SNR_i \right)$
- 6: Select top $S \times L$ nodes by priority
- 7: Randomly select k RSU nodes as initial center
- 8: **repeat**
- 9: Compute $score \leftarrow a_1 \left(f_{pi} / \sum_{Exepa} f_{pi} \right) + a_2 \left(SNR_i / \sum_{Exepa} SNR_i \right)$
- 10: Greedy assign nodes to groups according to $score$, ensuring each group at less 1 RSU
- 11: Update group centers
- 12: **until** center movement < tolerance
- 13: Compute group $score$
- 14: **if** score variance > 10% of mean, **then**
- 15: Swap nodes between highest/lowest score groups
- 16: **endif**
- 17: **for** $g = 1$ to S **do**
- 18: Calculate communication rate
- 19: Execute Hotstuff and compute consensus energy consumption E_{pag}^{HB} , E_{RSUg}^{HB}
- 20: **endfor**
- 21: Execute BLSPBFT and compute consensus energy consumption E_{RSU}^{PB}
- 22: $E_{pa}^B \leftarrow \sum_{g=1}^S E_{pag}^{HB}$, $E_{RSU}^B \leftarrow \sum_{g=1}^S E_{RSUg}^{HB} + E_{RSU}^{PB}$

5.2. Game-Based DRL Algorithm

For solving the optimization problems P1 and P2, a game-based DRL model is proposed in this section, in which the user acts as the leader and the RSUs as followers. First, the leader determines the offloading ratio $\varepsilon_{RSU_i}(t)$ via the agent. Then, the followers set their prices according to the user's offloading ratio to maximize their respective benefits. The specific game process of the SAC agent is illustrated in Figure 3.

**Figure 3.** The process of game-based DRL model.

5.2.1. State Space

In slot t , the system state includes the energy consumption and utility generated by RSUs during task processing, as well as the user's utility and the overall system utility. Let $E_{RSU}(t) = \sum_{i=1}^U (E_{RSUi}(t) + E_{RSU}^B(t)/U)$ denote the energy consumption of RSUs, and $U_{RSU}(t) = \sum_{i=1}^U U_{RSUi}(t)$ be RSU total utility. The user's utility and the total system utility are $U_{user}(t) = \sum_{i=1}^U U_{useri}(t)$ and $U_{all}(t) = U_{user}(t) + U_{RSU}(t)$, respectively. Accordingly, the state spaces of the leader and followers are $s_t^L = \{U_{user}(t), U_{all}(t)\}$ and $s_t^F = \{E_{RSU}(t), E_{pa}^B(t), U_{RSU}(t)\}$, and the system state space is $s_t = \{s_t^L, s_t^F\}$.

5.2.2. Action Space

During time slot t , the agent needs to determine the offloading ratio, the prices set by RSUs, the number of shards in the lower-layer blockchain, and the number of nodes within each shard. Therefore, the system's action space can be formulated as $a_t = \{a_t^L, a_t^F\}$, where $a_t^L = \{\varepsilon_{RSUi}(t)\}$ represents the leader's action decision space and $a_t^F = \{p_{RSUi}(t), S(t), L(t)\}$ denotes the action space of the followers.

5.2.3. Reward Function

In DRL, the agent obtains a reward after taking actions according to the environment. The rewards of the leader and followers are constructed by their respective utility functions, and a global reward function is adopted to balance the rewards of both sides. Since the magnitudes of different utility functions differ, normalization is required in the reward function design. Therefore, this paper uses the function $F(x) = \log(1+x)/\log(2+x)$ to normalize the reward functions.

Global reward function: To achieve a balance between the reward functions of the leader and followers, the global reward function is formulated as

$$r_t^G = F(U_{all}(t)) \quad (22)$$

Leader reward function: The leader agent handles the decision of the user offloading ratio, and its reward function is composed of the user's utility function. Thus, the reward function can be designed as

$$r_t^L = F(U_{user}(t)) + r_t^G \quad (23)$$

Follower reward function: Similar to the leader agent, the follower agent makes optimal decisions on the computation process and consensus process of RSUs. Thus, the reward function can be expressed as

$$r_t^F = F(U_{RSU}(t)) + r_t^G \quad (24)$$

5.2.4. Algorithm Design

This paper adopts SAC to construct the game-based DRL model. As an off-policy reinforcement learning (RL) method, the SAC algorithm improves the stability of the policy through the maximum entropy objective and makes more efficient use of sample data to accelerate the learning process. The objective of SAC is to find the optimal policy, i.e.,

$$\pi^* = \operatorname{argmax}_{\pi} \sum_{t=0}^T E_{(s_t, a_t) \sim \rho_{\pi}} [\gamma^t r_t + \alpha H(\pi(\cdot | s_t))] \quad (25)$$

In the above expression, ρ_π stands for the trajectory distribution probability under policy π , r_t is the reward function corresponding to the leader or follower, γ is the discount factor, α is the weight coefficient for entropy control, and $H(\pi(\cdot|s_t))$ represents the policy π entropy at s_t . To address the high-dimensional continuous control challenge, the SAC agent utilizes neural networks to approximate the policy function $\pi_\phi(a_t|s_t)$, soft Q-function $Q_\phi(s_t, a_t)$, and state value function $V_\phi(s_t)$. The objective function of the state value function $V_\phi(s_t)$ is given by:

$$J_V(\phi) = E_{s_t:D} \left[\frac{1}{2} \left(V_\phi(s_t) - E_{a_t:\pi_f} \left[Q_\chi(s_t, a_t) - \log \pi_f(a_t|s_t) \right] \right)^2 \right] \quad (26)$$

In the above equation, D denotes the replay buffer of historical experiences, and ϕ, χ represent the parameters of each neural network. Unlike traditional Q-learning counterparts, the soft Q-function incorporates an entropy regularization term, and its objective function is expressed as:

$$J_Q(\chi) = E_{(s_t, a_t, s_{t+1}):D} \left[\frac{1}{2} (Q_\chi(s_t, a_t) - Q_{target}(s_t, a_t))^2 \right] \quad (27)$$

Q_{target} can be expressed as

$$Q_{target}(s_t, a_t) = r_t + \gamma \left(Q_{\chi^-}(s_{t+1}, a_{t+1}) - \alpha \log(\pi_f(a_{t+1}|s_{t+1})) \right) \quad (28)$$

where χ^- represents the target Q-network weight parameters. The policy function is updated via the soft Q-function, with the expectation that the action probability distribution of the policy at state s_t matches the Q-value distribution. Accordingly, the update of the policy function is given by the following formula:

$$\begin{cases} J_\pi(\phi) = E_{s_t \sim D, \delta \sim N} [\log \pi_\phi(f_\phi(\delta; s_t)|s_t) - Q_\chi(s_t, f_\phi(\delta; s_t))] \\ \nabla_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(a_t|s_t) + (\nabla_{a_t} \log \pi_\phi(a_t|s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\phi f_\phi(\delta; s_t) \end{cases} \quad (29)$$

In the above equation, the function f calculates the mean and variance of the Gaussian distribution, and δ is the Gaussian sampling noise. Based on the SAC algorithm described above, we design a game-based DRL algorithm to solve the optimization problems P1–P2, as detailed in Algorithm 2.

Algorithm 2: Game-based DRL Algorithm

Input: Leader state s_t^L ; Follower state s_t^F ; Episode count N_k ; Step count N_t .

Output: Optimized policy for task offloading ε_{RSUii} ; price of RSU p_{RSUii} ;

Groups number S ; intra-group consensus nodes L .

- 1: Initialize leader and follower SAC networks parameters;
 - 2: Initialize replay buffers B_L, B_F for leader and follower respectively.
 - 3: **for episode = 1 to** N_k
 - 4: Reset environment and obtain initial states s_0^L, s_0^F .
 - 5: **for t = 1 to** N_t
 - 6: Leader agent observes state s_t^L and selects action a_t^L
 - 7: Follower agent observes its state s_t^F and leader's action a_t^L , then selects action a_t^F
 - 8: Execute joint action (a_t^L, a_t^F) at in the environment
 - 9: Observe next states (s_{t+1}^L, s_{t+1}^F) and immediate rewards (r_t^L, r_t^F)
 - 10: Store transition $(s_t^L, a_t^L, r_t^L, s_{t+1}^L)$ in replay buffer B_L
 - 11: Store transition $(s_t^F, a_t^F, r_t^F, s_{t+1}^F)$ in replay buffer B_F
-

Algorithm 2: *Cont.*

```

12:      Update SAC networks for both leader and follower using random batches
      sampled from their respective replay buffers
13:      Update target networks using soft update method
14: endfor
15: endfor

```

In Algorithm 2, the number of user vehicles is U , and the hidden layer sizes of the policy and value networks are fixed. Hence, action generation has a complexity of $O(U)$. Each training step also incurs $O(U)$ complexity. Let N_t be the number of training steps per time interval; then, the overall time complexity of Algorithm 2 is $O(N_t \times U)$.

5.3. Security Analysis

5.3.1. Analysis of the Attack Model

Adversary assumption: The adversary is assumed to be capable of controlling no more than $1/3$ of the consensus nodes in any shard of the lower-layer blockchain, and of launching flooding attacks against shards with poor load capacity to cause processing congestion, or of mounting cross-shard double-spending attacks.

- (1) **Shard flooding attack:** An adversary may flood a poorly performing shard with numerous invalid transactions to cause congestion and degrade system throughput. Nevertheless, the proposed scheme employs an improved K-Means algorithm to group the lower-layer consensus nodes, thereby balancing the processing performance across shards, while the upper-layer blockchain coordinates these groups. Consequently, the adversary is unable to sustain concentrated attacks on any single shard to create congestion.
- (2) **Cross-shard double-spending attack:** The adversary may initiate a transaction in one shard and concurrently use the same resource to launch another transaction in a different shard. In the two-layer blockchain, the lower layer handles only local transactions, whereas cross-shard transactions are confirmed via the upper-layer consensus. Thus, any conflicting cross-shard transaction is identified and denied by the upper layer, effectively thwarting cross-shard double-spending attacks.

5.3.2. System Security

This paper proposes a two-layer blockchain with parked vehicles and RSUs as consensus nodes. The lower layer uses improved K-Means for parallel grouping, boosting transaction efficiency. The upper layer coordinates these groups, ensuring transaction scalability. Hence, the security analysis of our scheme is summarized below.

- (1) **Efficient operation:** In the lower blockchain, RSUs act as leaders and parked vehicles as replicas to run Hotstuff consensus. Consensus nodes are sharded. All shards operate in parallel; each group handles task offloading information and transactions from user vehicles in its designated area and returns the results accordingly, enabling efficient task offloading.
- (2) **Computation verification:** Each shard processes its users' computing tasks under the coordination of the upper-layer blockchain, which also stores the final results. When a malicious vehicle uploads misleading results, vehicles and edge nodes can verify them by querying the upper-layer blockchain's transaction records.
- (3) **Scalability:** The lower-layer blockchain is sharded, with the upper layer coordinating inter-group transactions. Since the lower layer relies mainly on parked vehicles and requires few RSUs, the system achieves VEC transaction scalability with minimal RSUs.

- (4) Traceability: User tasks are audited by shard consensus nodes during offloading. Once block formation is complete, transactions become tamper-proof and are fully stored on upper-layer nodes. Law enforcement agencies can thus query the records to trace any user's history.

6. Experimental Results and Discussion

In this section, numerical simulations of the proposed scheme in this paper are conducted via MATLAB R2024B, and the system performance is evaluated. In the simulation, the PV parking duration data is from ACT Government Open Data Portal DataACT [46]. For the leader SAC, we set the policy network learning rate to 2×10^{-4} , the discount factor to 0.98, the mini-batch size to 128, and the replay buffer size to 106. And the value network learning rate is set to 2×10^{-4} . For the follower SAC agents, the learning rate of the policy network was set to 8×10^{-5} , the discount factor to 0.98, the mini-batch size to 128, the replay buffer size to 5×10^5 , and the learning rate of the value network to 2×10^{-4} . In addition, the experiment involved 3000 training episodes, each consisting of 50 steps. The resulting model size was less than 1 MB, making it suitable for deployment on resource-constrained devices. The key simulation parameters are presented in Table 2.

Table 2. Key simulation parameters.

Parameters	Value	Parameters	Value
$C_{vi}^{exe}, C_{rj}^{exe}$	24 cycles/bit [21]	d_0	100 m [43]
ω	1 KB	W_b	15 MB
D_{qi}	10–30 MB	θ	10×10^6 cycles [21]
D_{Bv}	4 MB	β	1×10^6 cycles [21]
f_{vi}	1–2.5 GHz	α	2 [43]
f_{rj}	4–6 GHz	η	1.63726×10^{-9} [43]
N_0	1.2589×10^{-13} W [43]	κ_v	10^{-27}
P_t	0.28183815 W [43]	κ_r	10^{-28}

First, the impact of the user offloading ratio ϵ_{RSUi} to RSUs on each utility function is investigated, as depicted in Figure 4. As observed from the figure, the required computational task load increases with the number of users. This trend elevates the utility of both users and RSUs, which consequently leads to a corresponding growth in the total utility. On the other hand, as ϵ_{RSUi} increases, a larger proportion of tasks are offloaded to RSUs. Consequently, the utility of RSUs gradually increases, while that of users decreases moderately. In addition, with a growing number of tasks offloaded to RSUs, high-performance edge servers substantially shorten the task execution time, which contributes to an increase in total utility.

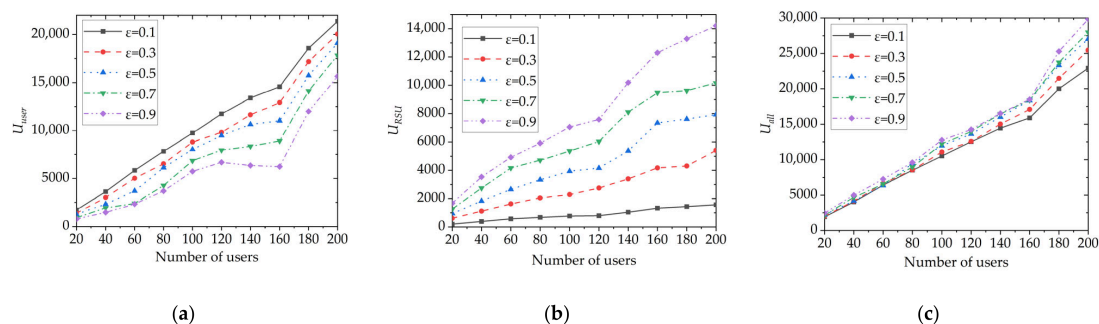


Figure 4. Influence of the user offloading ratio to RSU on utility function: (a) the influence of the user utility function; (b) the influence of the RSU utility function; (c) the influence of the total utility function.

Next, we compare our scheme with other blockchain structures, as illustrated in Figure 5. Figure 5a focuses on the comparison with single-layer blockchain architectures. Here, the single-layer PBFT is the scheme from [26], and comparisons are also made with single-layer BLSPBFT and Hotstuff. For clarity, the vertical axis is log-scaled. The figure reveals that with the two-layer architecture adopted in this paper, the lower layer is sharded, while the upper-layer blockchain serves as a coordinator. Furthermore, the average throughputs of single-layer PBFT, BLSPBFT, Hotstuff, and the proposed scheme are measured as 47.486016, 56.062086, 84.583428, and 4360.457961, respectively, indicating that the proposed scheme outperforms the other schemes with improvements of 0.989109856, 0.987143074, and 0.980602169, respectively. This leads to a substantial throughput improvement over the single-layer blockchain architecture. Figure 5(b) illustrates the impact on throughput of the two-layer blockchain structure formed by traditional consortium blockchain consensus mechanisms. In this comparison, the two-layer PBFT and two-layer BLSPBFT correspond to the schemes in refs. [47] and [48], respectively. As shown in the figure, the proposed scheme improves the throughput by 0.330415607, 0.232449872, and 0.136648142 over dual-layer PBFT, BLSPBFT, and Hotstuff, respectively. By considering performance balance among groups during sharding, the proposed scheme eliminates the adverse effect of low-performance groups on consensus, thus enhancing throughput. Figure 5c illustrates the impact on total system utility of the two-layer blockchain structure. The total utility is defined as the sum of user utility and RSU utility. Since PBFT and BLSPBFT adopt many-to-many communication transmission, Hotstuff exhibits evident advantages as a consensus algorithm for wireless communication nodes. In terms of signature, BLSPBFT and Hotstuff employ aggregate signatures, thereby achieving better computational performance than PBFT. Our scheme is designed by leveraging the respective advantages of BLSPBFT and Hotstuff. As can be observed from the figure, our scheme achieves the highest total utility, followed by the double Hotstuff scheme. Since double BLSPBFT and PBFT differ only in the digital signature method, their utility functions are relatively close.

Then, under the limited number of consensus nodes, different groupings of consensus nodes result in different numbers of consensus nodes within each group, thereby leading to different consensus performance. Furthermore, when the number of groups is fixed, consensus performance also varies with the node scale. Therefore, Figure 6 studies the utility function under different numbers of groups and different node scales. Specifically, Figure 6a fixes the total consensus node scale at 50 and evaluates throughput variation with different group numbers. Figure 6b fixes the group number at 6 and evaluates throughput variation with different node scales. As illustrated in Figure 6a, a rise in the number of users increases the task offloading per user, resulting in a gradual throughput decrease. Conversely, a higher number of groups implies more sharding, which substantially enhances transaction processing capacity and leads to a marked increase in throughput. Figure 6b shows that as the node scale increases, more consensus nodes are added to each shard, leading to longer consensus time and thus a gradual decrease in throughput.

Furthermore, the consensus node grouping scheme proposed in this paper is compared with the K-Means, K-Medoids, and random grouping schemes, as illustrated in Figure 7. Observations demonstrate that the proposed scheme achieves balanced allocation of computing and communication capabilities among nodes in different groups. Accordingly, the consensus energy consumption and latency of the lower-layer blockchain are reduced, and the consensus performance of each group is improved. Therefore, the utility function of our scheme outperforms that of the other benchmark schemes.

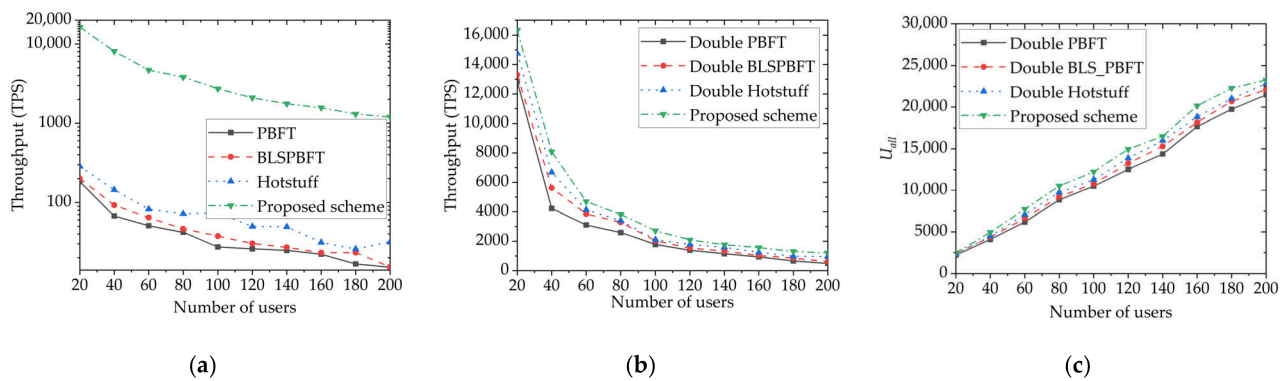


Figure 5. The influence of different blockchain schemes for system performance: (a) compare with a single blockchain on throughput; (b) compare with a double blockchain on throughput; (c) compare with a double blockchain on utility function.

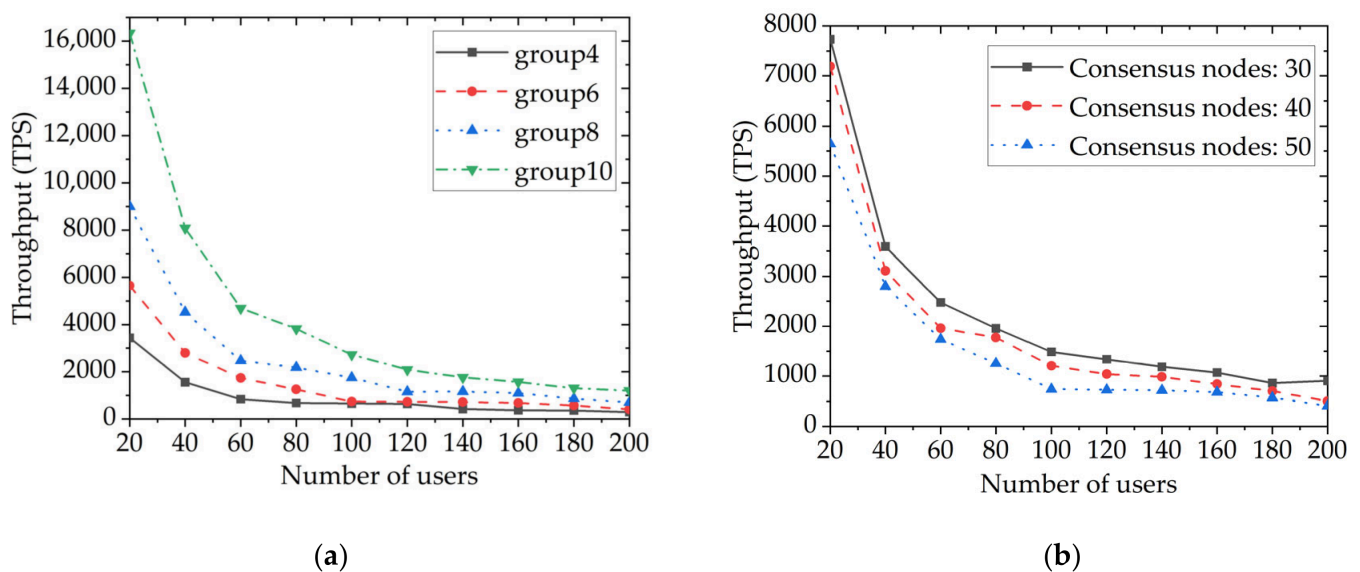


Figure 6. The variation in throughput for different group numbers and node scales: (a) the variation in throughput for different group numbers; (b) the variation in throughput for different node scales.

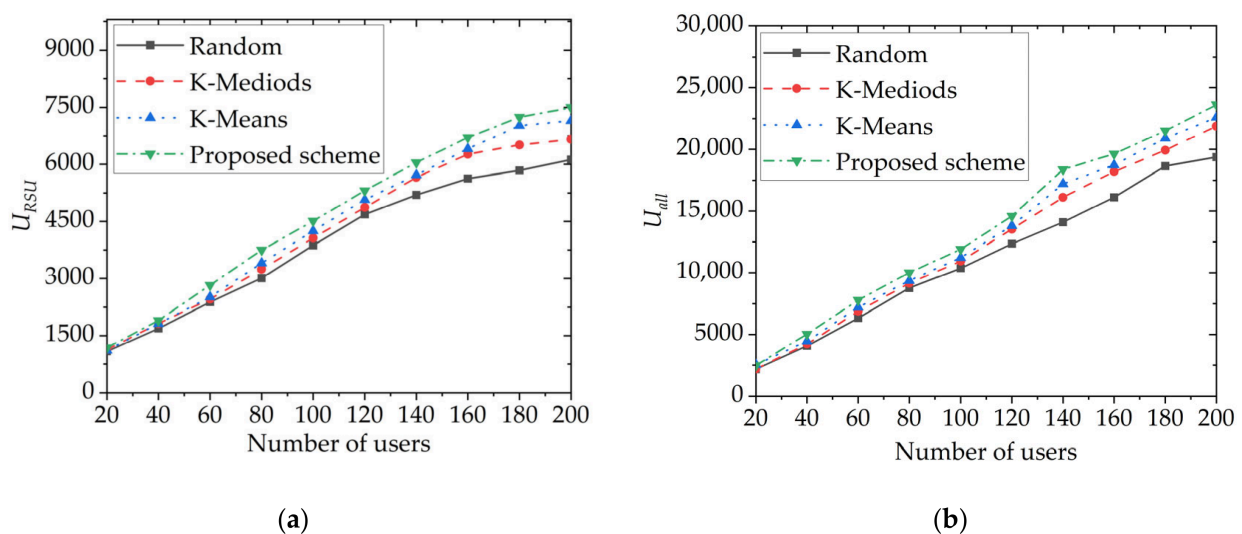


Figure 7. Variation in utility function with the number of users for different grouping methods: (a) variation in RSU utility function; (b) variation in total utility functions.

Next, a comparison between the proposed game-based DRL algorithm and other DRL methods is presented in Figure 8. In the figure, the proposed scheme constructs a game-based DRL framework by taking users as leaders and RSUs as followers. Consequently, the rewards of each agent are relatively balanced. As can be observed from Figure 8a, our scheme achieves the highest total utility function, which demonstrates that the agent game process does not impair the improvement of the overall reward. As can be observed from Figure 8b,c, the benchmark algorithms focus more on improving the utility of users, while our proposed scheme attains more balanced performance across all agents.

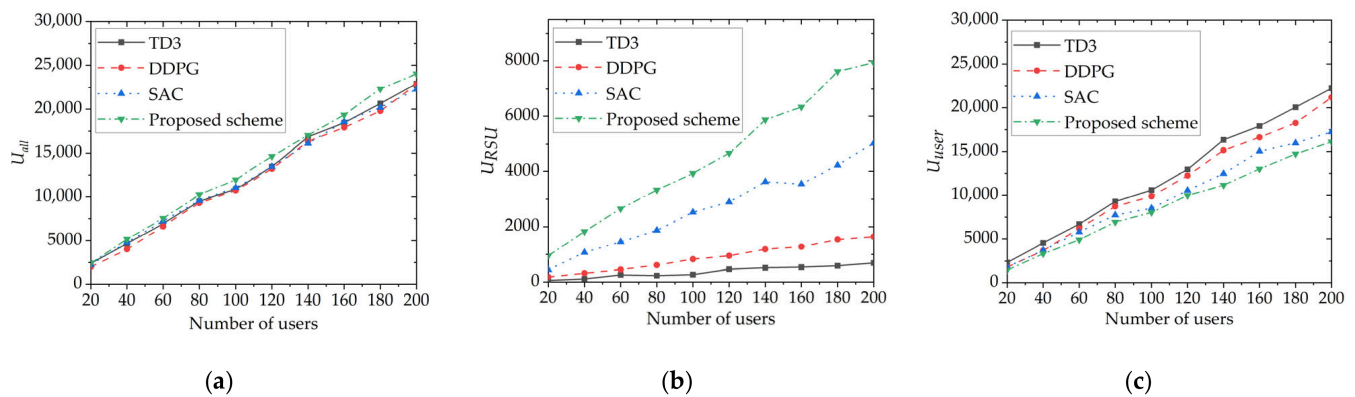


Figure 8. The influence of different DRL algorithms on utility functions: (a) total utility function; (b) RSU utility function; (c) user utility function.

Finally, we perform ablation experiments, as shown in Figure 9, where non-grouping algorithm, single-layer blockchain, and non-game DRL denote the system utility achieved without the grouping algorithm, without the dual-layer blockchain, and without the game DRL model, respectively. Notably, since the improved K-Means algorithm serves to balance the performance across groups, its removal exerts the most significant impact on the system utility. In contrast, the removal of the upper-layer blockchain and the exclusion of the game DRL model both affect the system utility to varying degrees, which demonstrates that the proposed scheme can effectively enhance system performance.

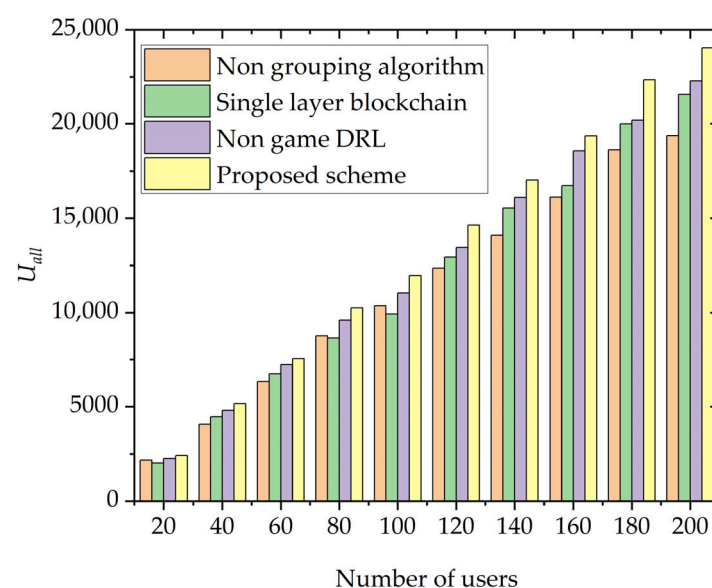


Figure 9. The influence of different components on the system's utility.

7. Conclusions

In this paper, a two-layer blockchain-based task offloading framework for VEC is proposed to improve the scalability of task offloading transactions. The framework employs Hotstuff as the consensus mechanism for the lower-layer blockchain and BLSPBFT for the upper-layer blockchain, thereby constructing a Hotstuff–BLSPBFT two-layer blockchain structure. For the lower layer, the improved K-Means algorithm is adopted to optimize the grouping of consensus nodes. Subsequently, utility functions of users and RSUs are formulated. A game-based DRL model is further established, where the user acts as the leader SAC agent and RSUs serve as the follower SAC agent, to optimize the utility functions, select the optimal offloading strategy and pricing, and determine the appropriate number of blockchain groups and nodes per group. Experimental results demonstrate that the proposed scheme can effectively guarantee the security of task offloading while significantly improving system efficiency. This paper is still insufficient in terms of large-scale node deployment and highly dynamic vehicular environments. Hence, future efforts will be directed toward secure task offloading in VEC under highly dynamic mobile scenarios, and toward scaling up the node size for improved large-scale deployment.

Author Contributions: Conceptualization, G.L. and F.Z.; methodology, G.L. and C.L.; software, G.L. and Z.S.; validation, M.C., C.L. and F.Z.; data curation, Z.S. and M.C.; supervision, F.Z. and C.L.; writing—original draft preparation, G.L. and Z.S.; writing—review and editing, M.C., C.L. and F.Z.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 62362013 and was funded by Guangxi Science and Technology Plan Project, grant number Guike AB25069120.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Li, C.; Xiao, K.; Xu, J.; Ji, W. Parked-vehicle-assisted task offloading in vehicular edge computing: A stackelberg game approach. *Comput. Netw.* **2026**, *274*, 111800. [[CrossRef](#)]
2. Wu, H.; Sun, Y.; Wolter, K. Energy-Efficient Decision Making for Mobile Cloud Offloading. *IEEE Trans. Cloud Comput.* **2020**, *8*, 570–584. [[CrossRef](#)]
3. Yang, L.; Liu, B.; Cao, J.; Sahni, Y.; Wang, Z. Joint Computation Partitioning and Resource Allocation for Latency Sensitive Applications in Mobile Edge Clouds. *IEEE Trans. Serv. Comput.* **2021**, *14*, 1439–1452. [[CrossRef](#)]
4. Peng, C.; Huang, X.; Wu, Y.; Kang, J. Constrained Multi-Objective Optimization for UAV-Enabled Mobile Edge Computing: Offloading Optimization and Path Planning. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 861–865. [[CrossRef](#)]
5. Chen, C.; Zeng, Y.; Li, H.; Liu, Y.; Wan, S. A Multihop Task Offloading Decision Model in MEC-Enabled Internet of Vehicles. *IEEE Internet Things J.* **2023**, *10*, 3215–3230. [[CrossRef](#)]
6. Xu, X.; Zhang, X.; Liu, X.; Jiang, J.; Qi, L.; Bhuiyan, M.Z.A. Adaptive Computation Offloading with Edge for 5G-Envisioned Internet of Connected Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 5213–5222. [[CrossRef](#)]
7. Zhou, Y.; Chai, Z.; Li, Y.; Li, J. Parking Vehicle-Assisted Task Offloading in Edge Computing: A dynamic multi-objective evolutionary algorithm with multi-strategy fusion response. *Swarm Evol. Comput.* **2025**, *94*, 101900. [[CrossRef](#)]
8. Shen, Q.; Hu, B.J.; Xia, E. Dependency-Aware Task Offloading and Service Caching in Vehicular Edge Computing. *IEEE Trans. Veh. Technol.* **2022**, *71*, 13182–13197. [[CrossRef](#)]
9. Liu, Y.; Wang, S.; Zhao, Q.; Du, S.; Zhou, A.; Ma, X.; Yang, F. Dependency-Aware Task Scheduling in Vehicular Edge Computing. *IEEE Internet Things J.* **2020**, *7*, 4961–4971. [[CrossRef](#)]
10. Tang, X.; Lou, L.; Peng, R.; Zhang, T.; Liu, Z.; Liu, Z.; Kang, J.; Han, Z.; Niyato, D. A Verifiable Privacy-Preserving Cross-Chain Protocol for Trusted Vehicle Edge Computing. *IEEE Trans. Veh. Technol.* **2026**, *75*, 7134–7149. [[CrossRef](#)]
11. Meng, X.; Liu, B.; Meng, X.; Liang, Y.; Deng, H. A Lightweight Group Authentication Protocol for Blockchain-Based Vehicular Edge Computing Networks. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 8556–8567. [[CrossRef](#)]

12. Liu, Q.; Gong, J.; Liu, Q. Blockchain-Assisted Reputation Management Scheme for Internet of Vehicles. *Sensors* **2023**, *23*, 4624. [[CrossRef](#)] [[PubMed](#)]
13. Chen, Q.; Li, C.; Chen, M.; Wu, M.; Zhang, G. Digital twin assisted multi-task offloading for vehicular edge computing under SAGIN with blockchain. *IET Commun.* **2025**, *19*, e70002. [[CrossRef](#)]
14. Lu, X.; Xiao, L.; Xiao, Y.; Xiong, Z.; Liu, Z.; Zhang, Y.; Zhuang, W. Blockchain-Enabled Secure Offloading for VEC: A Multi-Agent Reinforcement Learning Approach. *IEEE Trans. Dependable Secur. Comput.* **2025**, *22*, 2978–2995. [[CrossRef](#)]
15. Huang, X.; Ye, D.; Yu, R.; Shu, L. Securing parked vehicle assisted fog computing with blockchain and optimal smart contract design. *IEEE/CAA J. Autom. Sin.* **2020**, *7*, 426–441. [[CrossRef](#)]
16. Sarika, S.; Prabakeran, S. Blockchain-Based Multiobjective Secure Task Offloading Strategy Utilizing Optimized Conditional Self-Attention Generative Adversarial Network for Mobile Edge Computing in Internet of Vehicle. *Int. J. Commun. Syst.* **2025**, *38*, e70121. [[CrossRef](#)]
17. Liang, F. Decentralized and Network-Aware Task Offloading for Smart Transportation via Blockchain. *Sensors* **2025**, *25*, 5555. [[CrossRef](#)] [[PubMed](#)]
18. Fardad, M.; Muntean, G.; Tal, I. A Blockchain-Enabled Vehicular Edge Computing Framework for Secure Performance-Oriented V2X Service Delivery. *IEEE Trans. Veh. Technol.* **2024**, *73*, 13853–13867. [[CrossRef](#)]
19. Zhang, C.; Ji, M.; Wu, Q.; Fan, P.; Fan, Q. Optimizing System Latency for Blockchain-Encrypted Edge Computing in Internet of Vehicles. *Comput. Mater. Contin.* **2025**, *83*, 3519–3536. [[CrossRef](#)]
20. He, L.; Li, F.; Xu, H.; Xia, W.; Zhang, X.; Tao, X. Blockchain-based vehicular edge computing networks: The communication perspective. *Sci. China Inf. Sci.* **2023**, *66*, 172301. [[CrossRef](#)]
21. Lang, P.; Tian, D.; Duan, X.; Zhou, J.; Sheng, Z.; Leung, V.C.M. Blockchain-Based Cooperative Computation Offloading and Secure Handover in Vehicular Edge Computing Networks. *IEEE Trans. Intell. Veh.* **2023**, *8*, 3839–3853. [[CrossRef](#)]
22. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Poor, H.V. Cooperative Task Offloading and Block Mining in Blockchain-Based Edge Computing with Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2023**, *22*, 2021–2037. [[CrossRef](#)]
23. Zhang, D.; Shi, W.; St-Hilaire, M. The Permissioned Blockchain-Based Quantum-Inspired Edge Intelligence Approach for the Services of Future Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 6171–6185. [[CrossRef](#)]
24. Cao, Y.; Zhao, C.; Zhang, Y.; Jin, Y. Optimizing Resource Allocation and Energy Efficiency in Vehicle Mobile-Edge Computing with Blockchain Integration. *IEEE Internet Things J.* **2025**, *12*, 36807–36818. [[CrossRef](#)]
25. Wu, W.; Sun, D.; Jin, K.; Sun, Y.; Si, P. Proximal policy optimization-based committee selection algorithm in blockchain-enabled mobile edge computing systems. *China Commun.* **2022**, *19*, 50–65. [[CrossRef](#)]
26. Shi, J.; Du, J.; Shen, Y.; Wang, J.; Yuan, J.; Han, Z. DRL-Based V2V Computation Offloading for Blockchain-Enabled Vehicular Networks. *IEEE Trans. Mob. Comput.* **2023**, *22*, 3882–3897. [[CrossRef](#)]
27. Huang, Y.; Chen, Y.; Zhou, L.; Ren, J.; Li, H.; Wang, X.; Ning, Z. Blockchain Intelligence Empowered Task Offloading and Resource Trading for Secure Low-Altitude Economy Networking. *IEEE Trans. Netw. Sci. Eng.* **2026**, *13*, 6470–6487. [[CrossRef](#)]
28. Bai, F.; Shen, T.; Yu, Z.; Zeng, K.; Gong, B. Trustworthy Blockchain-Empowered Collaborative Edge Computing-as-a-Service Scheduling and Data Sharing in the IIoE. *IEEE Internet Things J.* **2022**, *9*, 14752–14766. [[CrossRef](#)]
29. Ren, J.; Li, J.; Liu, H.; Qin, T. Task offloading strategy with emergency handling and blockchain security in SDN-empowered and fog-assisted healthcare IoT. *Tsinghua Sci. Technol.* **2022**, *27*, 760–776. [[CrossRef](#)]
30. Li, W.; Sang, Y.; Jiang, T.; Badr, Y.; Liu, H. SLIM: Stateless-Based Lightweight Sharding Mechanism for Secure Data Interchange in Blockchain-Enabled Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2026**, *27*, 2225–2239. [[CrossRef](#)]
31. Chen, X.; Xue, G.; Yu, R.; Wu, H.; Wang, D. A Vehicular Trust Blockchain Framework with Scalable Byzantine Consensus. *IEEE Trans. Mob. Comput.* **2024**, *23*, 4440–4452. [[CrossRef](#)]
32. Zhang, Y.; Fan, W.; Yu, Y.; Liu, Y.A. DRL-Based Resource Orchestration for Vehicular Edge Computing with Multi-Edge and Multi-Vehicle Assistance. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 8764–8779. [[CrossRef](#)]
33. Shivanna, V.; Srikanth, A.; Tumuluru, V.K. Distributed Computation Offloading in Vehicular Edge Networks Using Proactive Service Reservations and Multi-Agent Learning. *IEEE Trans. Veh. Technol.* **2025**, *74*, 14564–14576. [[CrossRef](#)]
34. Xue, J.; Shao, F.; Zhang, T.; Tian, G.; Jiang, H. VEC Collaborative Task Offloading and Resource Allocation Based on Deep Reinforcement Learning Under Parking Assistance. *Wirel. Pers. Commun.* **2024**, *136*, 321–345. [[CrossRef](#)]
35. Xu, C.; Zhang, P.; Xia, X.; Kong, L.; Zeng, P.; Yu, H. Digital-Twin-Assisted Intelligent Secure Task Offloading and Caching in Blockchain-Based Vehicular Edge Computing Networks. *IEEE Internet Things J.* **2025**, *12*, 4128–4143. [[CrossRef](#)]
36. Wu, S.; Yang, L.; Li, J.; Guo, H.; Ahmad, I.; Da Costa, D.B.; Jiang, H.; Niyato, D. DRL-Based Pricing-Driven for Task Offloading and Dynamic Resource in Vehicle Edge Computing. *IEEE Trans. Mob. Comput.* **2025**, *24*, 10389–10404. [[CrossRef](#)]
37. Xue, J.; Yu, Q.; Wang, L.; Fan, C. Vehicle task offloading strategy based on DRL in communication and sensing scenarios. *Ad Hoc Netw.* **2024**, *159*, 103497. [[CrossRef](#)]

38. Jiang, H.; Guo, J.; Xiao, Z.; Yang, J.; Yang, K.; Min, G. Cooperative Content Caching in Vehicular Edge Computing Networks: A Two-Stage Deep Reinforcement Learning Approach. *IEEE Trans. Mob. Comput.* **2026**, *25*, 10557–10576. [[CrossRef](#)]
39. Ju, Y.; Cao, Z.; Li, M.; Liu, L.; Pei, Q.; Dong, M.; Mumtaz, S.; Guizani, M. NOMA and Hybrid Beamforming Aided Secure Computation Offloading for mmWave VEC Networks with Multi-Agent DRL. *IEEE Trans. Cogn. Commun. Netw.* **2026**, *12*, 6089–6103. [[CrossRef](#)]
40. Zhao, W.; Shi, K.; Liu, Z.; Wu, X.; Zheng, X.; Wei, L.; Kato, N. DRL Connects Lyapunov in Delay and Stability Optimization for Offloading Proactive Sensing Tasks of RSUs. *IEEE Trans. Mob. Comput.* **2024**, *23*, 7969–7982. [[CrossRef](#)]
41. Xu, X.; Liu, K.; Dai, P.; Jin, F.; Ren, H.; Zhan, C.; Guo, S. Joint task offloading and resource optimization in NOMA-based vehicular edge computing: A game-theoretic DRL approach. *J. Syst. Archit.* **2023**, *134*, 102780. [[CrossRef](#)]
42. Ma, X.; Zhao, J.; Wang, Y.; Zhang, T.; Li, Z. A New Approach to SINR-Based Reliability Analysis of IEEE 802.11 Broadcast Ad Hoc Networks. *IEEE Commun. Lett.* **2021**, *25*, 651–655. [[CrossRef](#)]
43. Ma, X.; Trivedi, K.S. SINR-Based Analysis of IEEE 802.11p/bd Broadcast VANETs for Safety Services. *IEEE Trans. Netw. Serv. Manag.* **2021**, *18*, 2672–2686. [[CrossRef](#)]
44. Liang, G.; Li, C.; Zhao, F.; Zhang, C.; Zhu, L. Task Offloading for Vehicular Edge Computing Based on Improved Hotstuff Under Parking Assistance. *IEEE Trans. Veh. Technol.* **2025**, *74*, 14550–14563. [[CrossRef](#)]
45. Ye, X.; Li, M.; Yu, F.R.; Si, P.; Wang, Z.; Zhang, Y. MEC and Blockchain-Enabled Energy-Efficient Internet of Vehicles Based on A3C Approach. In Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 7–11 December 2021; pp. 1–6.
46. Government, A. Smart Parking Stays Dataset. Available online: <https://www.data.gov.au/data/dataset/3vsj-zpk7> (accessed on 1 January 2023).
47. Wu, J.; Yang, R.; Li, M.; Sun, E.; Yu, F.R.; Zhang, D.; Si, P. Efficiency Optimization for Blockchain-Enabled V2V Energy Trading with Dynamic Clustering Based on Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2026**, *75*, 405–418. [[CrossRef](#)]
48. Liu, C.; Ning, K.; Mathiopoulos, P.T.; Xue, Z.; Han, G. A Novel Dual-Layer Multi-Shard Blockchain Architecture for Vehicle Data Sharing. *IEEE Trans. Mob. Comput.* **2026**, *25*, 9468–9486. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.