DRL-Based V2V Computation Offloading for Blockchain-Enabled Vehicular Networks

Jinming Shi, *Student Member, IEEE,* Jun Du, *Senior Member, IEEE,* Yuan Shen, *Member, IEEE,* Jian Wang, *Senior Member, IEEE,* Jian Yuan, *Member, IEEE* and Zhu Han, *Felllow, IEEE*

Abstract—Vehicular edge computing (VEC) is an effective method to increase the computing capability of vehicles, where vehicles share their idle computing resources with each other. However, due to the high mobility of vehicles, it is challenging to design an optimal task allocation policy that adapts to the dynamic vehicular environment. Further, vehicular computation offloading often occurs between unfamiliar vehicles, how to motivate vehicles to share their computing resources while guaranteeing the reliability of resource allocation in task offloading is one main challenge. In this paper, we propose a blockchain-enabled VEC framework to ensure the reliability and efficiency of vehicle-to-vehicle (V2V) task offloading. Specifically, we develop a deep reinforcement learning (DRL)-based computation offloading scheme for the smart contract of blockchain, where task vehicles can offload part of computation-intensive tasks to neighboring vehicles. To ensure the security and reliability in task offloading, we evaluate the reliability of vehicles in resource allocation by blockchain. Moreover, we propose an enhanced consensus algorithm based on practical Byzantine fault tolerance (PBFT), and design a consensus nodes selection algorithm to improve the efficiency of consensus and motivate base stations to improve reliability in task allocation. Simulation results validate the effectiveness of our proposed scheme for blockchain-enabled VEC.

Index Terms—Vehicular edge computing (VEC), computation offloading, blockchain, deep reinforcement learning (DRL).

1 Introduction

With the development of autonomous driving and Internet of Vehicles (IoV) technologies, various vehicular applications have been emerging to improve the traffic safety and driving experience. However, because of the limited onboard computing and storage resources, some computation-intensive tasks cannot be performed within the deadlines, which need the assistance of central cloud or edge computing [1], [2]. Due to the high transmission delay between vehicles and the central cloud, vehicular edge computing (VEC) is commonly exploited to improve the computing capability of vehicles, where vehicles can transmit computing tasks to either base station (BS) or neighboring vehicles. Considering that with the increase of the traffic density in the coverage of a BS, limited computing capability in the BS cannot satisfy all the vehicular computing requirements at the same time. Vehicle-to-vehicle (V2V) computation offloading is becoming a promising mechanism that can alleviate the workload of BS effectively, where task vehicles with limited computing resources can offload some computing tasks to neighboring vehicles with idle computing resources [3]. Moreover, compared to the fixed-position BS, the link duration between two vehicles driving in the same direction is longer because the relative velocity between the two vehicles is smaller.

However, most studies on VEC aimed at the computing resource allocation and the incentive of sharing computing resources among vehicles, the security and trust issues in VEC were seldom considered. Due to the high mobility of vehicles, V2V computation offloading always occurs between unfamiliar vehicles. Service vehicles may not allocate sufficient computing resources for task vehicles, so that the offloading tasks may not be completed within the maximum tolerable delay. Furthermore, in the process of V2V computation offloading, some traditional methods, such as game theory [4] and auction mechanism [5], [6], can only optimize the relationship between resource assignment and service price, the reliability of resource allocation in service vehicles is hardly guaranteed. Therefore, it is necessary for VEC to establish a mechanism that ensures the security and reliability in the process of vehicular computation offloading, and meanwhile motivates vehicles to share more computing resources to reduce the offloading delay and improve the completion ratio of vehicular offloading tasks.

To meet the requirements of security and reliability in computation offloading, some works [7], [8], [9] have applied blockchain in edge computing. In the scenario of VEC, blockchain can effectively improve the security and reliability of V2V computation offloading [10], where transactions in terms of vehicular computation offloading are recorded in a decentralized, transparent and immutable manner [11], [12]. Moreover, blockchain relies on data transmitted by the P2P network layer while vehicular edge computing (VEC) paradigm originates from P2P but extends the peer to vehicles, thus blockchain and VEC have the same distributed network framework [13], [14]. Besides, blockchain can be regarded as a distributed database, and blockchain consensus requires massive computation, while in VEC, vehicles can access edge servers for storage and computing services. Therefore, blockchain and VEC have the same functions

- *J. Shi, J. Du, Y. Shen, J. Wang, and J. Yuan are with the Department of Electronic Engineering, Tsinghua University, Beijing, 100084, China. E-mail: shijinmings@163.com, blgdujun@gmail.com, {shenyuan_ee, jianwang, jyuan}@tsinghua.edu.cn.*
- *Zhu Han is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77004 USA, and also with the Department of Computer Science and Engineering, Kyung Hee University, Seoul 446-701, South Korea. E-mail: zhan2@uh.edu.*

of storage and computation. It is worth noting that in a public blockchain, all distributed nodes need to participate in the consensus process, and it takes a long time in block generation and verification, which is not suitable for high dynamic vehicular networks. As a result, some works [15], [16] employed consortium blockchain in the VEC system, where only a part of BSs are authorized to participate in the consensus process of blockchain. To improve the efficiency of consensus, some works [17], [18] proposed non-compute-intensive algorithms based on practical Byzantine fault tolerance (PBFT), which are suitable for lightweight IoV-oriented blockchains.

Furthermore, there are still some challenges to be solved in blockchain-enabled VEC. First, due to the dynamic vehicular environment, it is challenging to establish an efficient mechanism for vehicular task offloading with the consideration of vehicle's mobility and heterogeneous vehicular computing capability. Second, how to motivate vehicles to contribute their onboard computing resources and meanwhile guarantee the reliability of onboard resource allocation is also a problem that should be carefully investigated. Third, the performance of PBFT-based consensus protocol in consortium blockchain depends on the number of consensus nodes and the block size. Therefore, how to design a scheme that jointly optimizes the consensus nodes selection and block size with the aim of improving the performance of blockchain is also a challenge.

Motivated by the challenges mentioned above, we design a VEC framework based on a lightweight blockchain, which makes V2V computation offloading performed in a secure, reliable, and efficient manner. Besides, considering the dynamic vehicular environment, the policy of vehicular task offloading should adapt to the state of the vehicular network, we thus utilize deep reinforcement learning (DRL) in vehicular task allocation, where the policy of task offloading is updated in real time by observing the state of the vehicular network. The main contributions of this paper are summarized as follows:

- We propose a blockchain-enabled VEC framework, where BSs are utilized to maintain a consortium blockchain. The blockchain is responsible for evaluating the reliability of vehicles according to historical transactions and further motivating vehicles to make appropriate resource allocation. Moreover, the smart contract in blockchain facilitates the sharing of vehicular computing resources on demand by automatically running the vehicular computation offloading algorithm, and guarantees the security and reliability of task offloading.
- We develop a smart contract-based vehicular task allocation scheme, where dynamic pricing and DRL are utilized to motivate vehicles to contribute their idle computing resources and improve the efficiency and reliability of V2V task offloading. Furthermore, to make the proposed scheme adapt to the dynamic vehicular environment, the vehicular computing capability, the V2V link state, and the reliability of service vehicles are all considered in task allocation.
- Since the reliability of service vehicles is evaluated by previous task offloading events recorded in the transactions, and the accuracy of the estimated reliability depends on the timeliness of the transactions, we design a PBFT-based consensus, and develop a DRL-based algorithm to improve the efficiency of transaction verification by jointly optimizing consensus nodes selection and block size, and meanwhile motivate BSs to improve reliability in task allocation.

The remainder of the paper is organized as follows. The related works are presented in Section 2. In Section 3, we present the architecture of the blockchain-enabled VEC. Section 4 describes the V2V computation offloading algorithm in smart contract. The consensus nodes selection in the blockchain is detailed in Section 5. Finally, we present the simulation results and conclusion in Section 6 and Section 7, respectively.

2 Related Work

2.1 Vehicular Edge Computing

The problem of vehicular task offloading in VEC has been widely investigated in recent works, and some optimization methods have been utilized to solve the problem. In [19], the task offloading problem was solved by a modified Ant Colony Optimization algorithm. In [5], the problem of computation offloading in VEC was formulated as an auction-based graph job allocation problem and solved by a structure-preserved matching algorithm. To minimize the overall response time of offloading tasks with interdependency, a modified genetic algorithm-based task scheduling scheme that considered the instability and heterogeneity of VEC was developed in [20]. Moreover, some learning-based methods were also proposed to solve the problem of vehicular task offloading. In [21], a DRL-based resource allocation framework with multi-timescale was proposed by considering the constraints of vehicle mobility and task delay in VEC. In [1], authors designed an imitation learning-enabled online task scheduling algorithm for VEC. Since executing offloading tasks occupies a vehicle's onboard computing resources and consumes the vehicle's energy, some vehicles may not be willing to execute offloading tasks. To motivate vehicles to contribute their idle computing resources, some incentive mechanisms were proposed in recent works. For example, authors in [22] designed a contract-based incentive mechanism that combined resource contribution and resource utilization, while in [23], a timeliness-aware incentive mechanism was proposed to stimulate the participation of vehicles with the consideration of the vehicle's uncertain travel time.

All of the works mentioned above treat the service providers as trustworthy devices and do not consider the security and privacy of task offloading in VEC. In this work, we exploit blockchain technology to guarantee security and reliability in vehicular computation offloading.

2.2 Blockchain for Vehicular Networks

Recently, the integration of blockchain and vehicular networks has been studied in some works, where blockchain is utilized to ensure the security of vehicular task offloading. In [24], blockchain and smart contract were employed to facilitate fair task offloading and mitigate security attacks in VEC. In [25], consortium blockchain was employed in a vehicular computing resource trading system to guarantee transaction security and privacy protection in a distributed manner. Authors in [26] proposed a resource transaction architecture based on blockchain to provide resource preallocation service for vehicles and eliminate reliance on third-party platforms. Moreover, to ensure the reliability of vehicles in task offloading, some works employed blockchain to store the social reputation of vehicles which is used for offloading. In [27], RSUs allocated computational tasks for nearby fog vehicles based on repute scores maintained by a semi-private consortium blockchain. In [15], a lightweight blockchain-based resource sharing scheme was proposed in VEC, where the reputation value was designed to indicate the trustworthy degree of vehicles in task offloading. Besides, to optimize the performance of blockchain-based VEC systems, some works also proposed various consensus algorithms to improve the efficiency of blockchain consensus. Authors in [18] exploited Byzantine fault tolerance-based Proof-of-Stake consensus to provide fast and deterministic block finalization, and provided a contract-based incentive mechanism to motivate vehicles to share their computing resources.

However, few of the works mentioned above investigated the dynamic trust management for both BSs and vehicles in VEC. In this paper, we propose a blockchain-enabled VEC framework that considers the reliability of both BSs and service vehicles in task allocation and task execution. We further develop a DRL-based algorithm combined with dynamic pricing for vehicular computation offloading, and design an enhanced consensus protocol to improve the performance of blockchain.

3 Blockchain-Enabled VEC

In this section, we will demonstrate the architecture of our proposed blockchain-enabled VEC system in Section 3.1 and Section 3.2, and further describe our proposed smart contract for vehicular computation offloading and the consensus scheme for the blockchain-enabled VEC in Section 3.3 and Section 3.4, respectively. For ease of reference, the key notations in this paper are summarized in Table 1.

3.1 VEC System Model

As shown in Fig. 1, we consider a VEC scenario where multiple BSs are distributed alongside the roads, and each BS is equipped with an edge server that is responsible for computing resource allocation among vehicles. When a vehicle enters the communication range of a BS and is willing to participate in the VEC, it first sends a registration request message that includes the ID, current position, velocity, and available onboard computing resources to the BS. When a vehicle cannot execute all of its local tasks due to the limited onboard computing resources, it can offload its computing tasks to either BSs or neighboring vehicles that drive in the same direction. Considering that with the number of task offloading requests increasing, the BS cannot handle a large number of tasks simultaneously due to the limited computing capability, some computing tasks should be offloaded to neighboring vehicles with idle computing resources. In this work, we will mainly focus on the problem of V2V computation offloading in VEC.

Since the position of BSs is stationary and BSs serve nearby vehicles all the time, BSs can be used to collect vehicular computing service requests, observe the states of the vehicles inside the communication range, and allocate

TABLE 1
Summary of Key Notations

Notation	Description
a	Cost of signing a block/transaction
b	Cost of verifying a signature
c	Cost of generating/verifying a MAC
N_c	Number of consensus nodes
f	Maximum number of faulty nodes
$S_{bk}, \dot{S}_{bk}$	Block size, maximum block size
M_t	Number of transactions in a block
χ	Average transaction size
T^I	Block interval
T^V	Time of block verification
T^F	Time-to-finality (TTF)
ϕ_n	The n^{th} offloading task
D_n	Data size of task ϕ_n
C_n	Computation size of task ϕ_n
τ_n	Maximum delay of task ϕ_n
t_n	Offloading delay of task ϕ_n
f_n	Allocated computing resources for task ϕ_n
p_n	Service price of task ϕ_n
U_n	Utility of ϕ_n
V_s	The s^{th} service vehicle
F_s	Available computing resources in V_s
η_s	Reliability of V_s
x_n^s	Decision variable of whether ϕ_n is offloaded to V_s
V_t	Task vehicle
$\mathcal{D}$	Communication range of a vehicle
$\mathcal{U}_n$	Utility of task vehicle for offloading ϕ_n
I_b	The b^{th} BS
G_b	Available computing resources in I_b
ι_b	Reliability of I_b
y_b	Decision variable of whether I_b is consensus node
N_{tr}	Number of transactions generated in a block interval

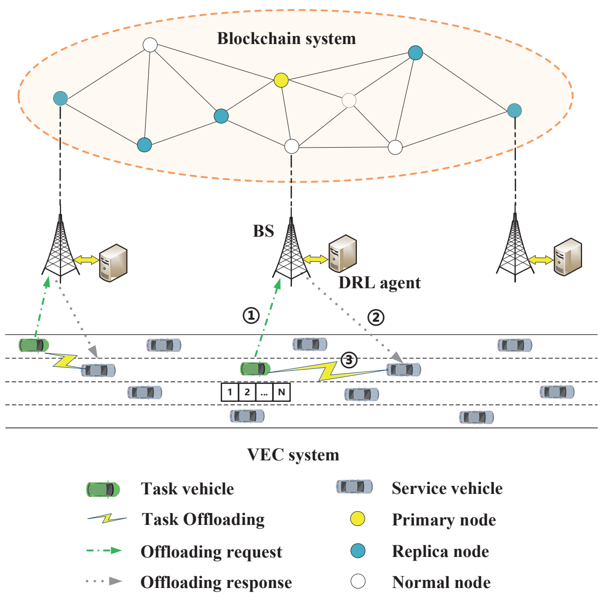


Fig. 1. The architecture of the blockchain-enabled VEC system.

vehicular computing tasks to vehicles with idle computing resources. If a vehicle needs to offload part of its computing tasks due to the limited onboard computing resources, it first sends a service request to the nearby BS, and we call this vehicle as the task vehicle, and call the vehicles in the communication range of the task vehicle as the service vehicles. Then, the BS performs the vehicular task allocation and selects some of the service vehicles to execute the computing tasks from the task vehicle. During the process of vehicular task allocation, the BS can collect massive experiences in terms of vehicular task offloading which can be used for training the DRL-based algorithm of vehicular task allocation. Moreover, to motivate service vehicles to contribute their idle computing resources, the task vehicle pays a service price for each offloading task, and the selected service vehicle allocates part of its computing resources for the offloading task according to the service price. Furthermore, BS can use the pre-trained DRL model and perform the algorithm of task allocation to dynamically select the service vehicles and the service price according to the states of vehicles in the communication range of the BS.

3.2 Blockchain Model

Due to the high dynamic vehicular environment, a task vehicle driving on the road may have different neighboring service vehicles in different time slots, and the task vehicle may not obtain much information from all of the neighboring vehicles in real time, which makes it difficult to evaluate the willingness of neighboring vehicles to contribute their computing resources. In some cases, a service vehicle agrees to accept the service request from the task vehicle, but the computing resources that the service vehicle allocates for the offloading task cannot ensure the task is completed within the deadline, which leads to an offloading failure. Therefore, it is necessary to construct an authentication mechanism for vehicles in VEC. Blockchain is deemed as a promising solution, where the historical transactions of vehicular task offloading are stored and utilized to evaluate the reliability of vehicles in resource allocation. Additionally, considering that public blockchain requires massive computation and time cost in the consensus process, which is not suitable for vehicular networks, we thus employ a lightweight consortium blockchain in the VEC system, where only a part of BSs are authorized to participate in the blockchain consensus. Compared to the public blockchain, there are fewer consensus nodes in the consortium blockchain, and the computation complexity of blockchain consensus is lower.

In the blockchain-enabled VEC, considering the fast-changing situation in vehicular networks due to the mobility of vehicles, the blockchain in the VEC system is maintained by stationary BSs, and the transactions in terms of vehicular task offloading are recorded by BSs as well. Moreover, the transactions are grouped and stored in a persistent, immutable, and tamper-proof ledger, and are verified by the consensus before attaching to the blockchain. Furthermore, the traceability of V2V computation offloading in blockchain makes it possible for guaranteeing the security and reliability of vehicles and BSs. Specifically, when a BS performs the vehicular task allocation, it first collects the information of vehicles in the communication range of the BS and finds the reliability of the vehicles by looking up the table of reliability values stored in the BS with vehicle IDs, and then selects the service vehicle and service price by a DRL-based task allocation algorithm. The table of reliability values is periodically updated according to the verified transactions in terms of vehicular task offloading. In addition, to motivate BSs to perform vehicular task allocation appropriately, we also evaluate the reliability of BSs according to the performance in vehicular task allocation and regard the reliability of BSs as a basis in the consensus nodes selection.

3.3 Smart Contract

In the blockchain-enabled VEC, the smart contract is a predefined script [10], which is utilized to perform the computing task allocation among vehicles and record the events of task offloading automatically. To incentivize vehicles to share their idle computing resources, a dynamic pricing scheme is exploited in vehicular task offloading, where service vehicles allocate their idle computing service according to the received service price. Furthermore, a DRL-based algorithm of vehicular task allocation is developed to dynamically select the service vehicle and determine the service price for each offloading task from task vehicles. Once a task vehicle sends a task offloading request to the BS, the smart contract in the BS processes the offloading request and triggers the DRL-based task allocation algorithm deployed in the BS. After the task is completed by the service vehicle, the BS verifies the execution result. If the result is valid and the offloading delay does not exceed the deadline of the task, the task vehicle sends the service price to the service vehicle, and a transaction that records the task offloading event is generated in the blockchain. The detailed algorithm of vehicular task allocation is presented in Section 4.

3.4 Consensus Process

After each offloading task is completed, the offloading information is recorded as a transaction in the blockchain system, which includes the IDs of the task vehicle and service vehicle, the task profile, the offloading delay, the execution result of the task, and the timestamp. In each round of blockchain consensus, newly generated transactions are verified by blockchain nodes and permanently stored in blocks, and the verified transactions can be used for evaluating the reliability of service vehicles and BSs. Considering that the accuracy of the estimated reliability depends on the timeliness of the transactions, the delay of transaction verification should be optimized in the blockchain consensus. We thus propose a consensus algorithm to improve the efficiency of transaction verification. In the following, we will demonstrate how to implement the consensus in the blockchain system.

Due to the high dynamic nature of vehicular networks, the consensus delay of the blockchain should be short enough so that the information of vehicular computation offloading can be updated in time. Some existing consensus protocols, such as Proof-of-Work (PoW), Proof-of-Stake (PoS), suffer from a long time in the consensus process, which is not suitable for the VEC scenario. Therefore, in the consensus of the blockchain-enabled VEC, we employ the PBFT protocol [28], [29], which has been widely applied

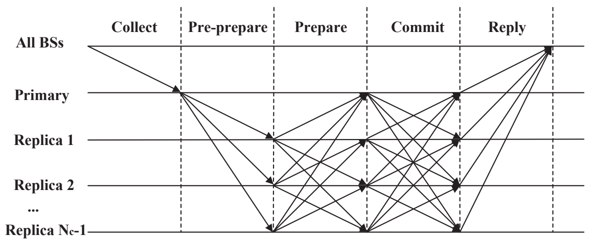


Fig. 2. The main process of the PBFT-based consensus.

in consortium blockchains and can achieve less consensus latency than PoW and PoS. In the traditional PBFT consensus algorithm [30], the consensus nodes are predetermined and the primary node is selected from the consensus nodes in round-robin order. Since the consensus performance is influenced by the number of consensus nodes, we then propose an enhanced consensus scheme based on PBFT to optimize the performance of blockchain consensus, where the primary node and replica nodes are dynamically selected from BSs in a certain area according to the state of BSs, and the block size is determined according to the state of BSs as well. In the consensus scheme, the primary node and replica node are responsible for block generation and block verification, respectively, and the unselected BSs act as the normal nodes, which only record task offloading results as transactions in the blockchain and do not participate in the block generation and verification. Additionally, the blockchain consensus employs signature and message authentication code (MAC) to ensure the integrity and authentication of a transaction. In the consensus, signing a block or transaction, verifying a signature, generating a MAC, and verifying a MAC require a, b, c, c CPU cycles [31], [32], respectively. Moreover, to guarantee the security of the blockchain system, the consensus allows at most $f = \lfloor (N_c - 1)/3 \rfloor$ consensus nodes are faulty [29]. Finally, the main steps of the consensus are shown as follows.

3.4.1 Consensus Nodes Selection

At the beginning of each round of consensus, the primary node in the previous round of consensus performs the algorithm of consensus nodes selection to select one primary node and multiple replica nodes from BSs in a given area, and the other BSs in the area are the normal nodes. The available computing resources for the consensus in the BS and the reliability of the BS in vehicular task allocation are the two main factors that affect the consensus nodes selection. In other words, the BS with higher reliability and more computing resources will have a higher probability of being selected as a consensus node. In each round of consensus, each consensus node will gain a reward for the contribution to the blockchain consensus. Therefore, to gain more rewards from the blockchain, each BS tends to improve its reliability in vehicular task allocation, which prevents some malicious BSs from giving inappropriate or unfair policies in vehicular task allocation. The detailed algorithm of consensus nodes selection is presented in Section 5.

3.4.2 Collect

At the beginning of a block interval, the primary node first collects unverified transactions from all BSs and sorts the transactions by timestamps. Then, the primary node forms the first M_t transactions as an unvalidated block, and the size of the block is denoted as S_{bk}. We assume that the average size of transactions is χ, then we have $M_t = \lfloor S_{bk}/\chi \rfloor$. Then, the primary node verifies M_t signatures and M_t MACs from the M_t transactions. The computation cost of primary node is $M_t(b + c)$.

3.4.3 Pre-prepare

In this step, the primary node generates one signature and one MAC for the unvalidated block and generates $N_c - 1$ MACs for the pre-prepare message. The pre-prepare message that contains the block is then sent to other replica nodes. Each Replica node verifies one MAC from the block, and M_t MACs and M_t signatures from the transactions in the block. Then, the computation cost of the primary node is $a + N_c c$, and the computation cost of each replica node is $c + M_t(b + c)$. In addition, without loss of generality, we assume that the transmission time of a block is proportional to the block size. Then, the time of message delivery is $\tau_{bk} S_{bk}$.

3.4.4 Prepare

Once verifying the pre-prepare message, replica nodes send the prepare message to other consensus nodes. After each replica node collects $2f$ prepare messages matched with the pre-prepare message, the consensus enters the next step. In this step, the primary node needs to verify $2f$ MACs, every replica node needs to generate $N_c - 1$ MACs and verify $2f$ MACs. Therefore, the computation cost of the primary node is $2fc$, and the computation cost of each replica node is $(N_c - 1 + 2f)c$. The time of message delivery is $\tau_{bk} S_{bk}$.

3.4.5 Commit

Upon receipt of $2f$ matching prepare messages, each consensus node sends a commit message to the other consensus nodes. In this step, the primary node and each replica node verify $2f$ MACs and generate $N_c - 1$ MACs. The computation cost of the primary node and each replica node is $(N_c - 1 + 2f)c$. The time of message delivery is $\tau_{bk} S_{bk}$.

3.4.6 Reply

Upon receipt of $2f$ matching commit messages, replica nodes accept the block as valid and append the verified block to its local ledger, and send the reply message containing the validated block to other BSs. Then, each BS updates its global view from the verified block after receiving $f + 1$ matching reply messages. In this step, the primary node and each replica node generate one MAC for the reply message, the computing cost of each node is c. The time of message delivery is $\tau_{bk} S_{bk}$.

The main process of consensus is shown in Fig. 2. Finally, the computation time of the primary node is

$$T_p^V = \frac{M_t(b + c) + a + (2N_c + 4f)c}{G_p}, \tag{1}$$

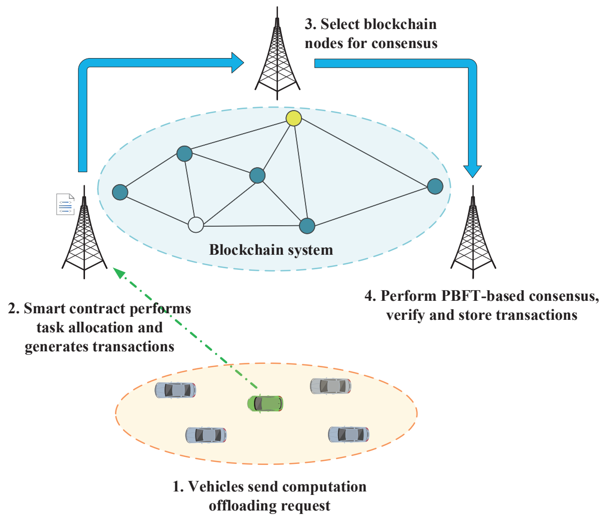

3. Select blockchain nodes for consensus
Blockchain system
2. Smart contract performs task allocation and generates transactions
4. Perform PBFT-based consensus, verify and store transactions
1. Vehicles send computation offloading request

Fig. 3. The main process of the blockchain-enabled VEC system.

where G_p is the available computing resources in the primary node. Similarly, the computation time of a replica node is

$$T_r^V = \frac{M_t(b+c) + (2N_c + 4f)c}{G_r}. \tag{2}$$

Similar to [33], we use *time to finality* (TTF) to represent the latency of the consensus process, which is shown as

$$T^F = T^I + T^D + T^V, \tag{3}$$

where T^I, T^D, and T^V denote the block interval, time of message delivery, and time of verification, respectively. According to the above analysis, we have $T^D = 4\tau_{bk}S_{bk}$, and $T^V = \max\{T_p^V, T_r^V\}$. Moreover, in order to prevent newly generated blocks suffering from long time of verification, the TTF should satisfy the following constraint:

$$T^F \leq \delta T^I, \tag{4}$$

where $\delta > 1$ is a constant. In (4), T^F contains block generating interval T^I and the time of block verification, and $T^F \leq \delta T^I$ ensures that the time of block verification should be less than a threshold so that newly generated blocks can be verified in each round of consensus, and the threshold is determined by T^I.

3.5 Process of Blockchain-Enabled VEC

The main process of our proposed blockchain-enabled VEC system is presented in Fig. 3. In the VEC system, a task vehicle that has several offloading tasks first sends the offloading request to the nearby BS, then the smart contract in the BS is triggered. The BS collects the information of the vehicles surrounding the task vehicle and evaluates the reliability of the vehicles according to the transactions stored in the consortium blockchain, and then selects a service vehicle for task offloading. After the V2V computation offloading is finished, a transaction that records the result of the computation offloading is generated and verified by the blockchain consensus. In each round of consensus, the consensus nodes are selected from the BSs in the VEC system and then perform the PBFT-based consensus. Finally, the verified transactions are stored in the blockchain system and can be further used for evaluating the reliability of vehicles and BSs.

4 Smart Contract for V2V Computation Offloading

In this section, we will detail our proposed smart contract for vehicular task allocation in the blockchain-enabled VEC system. In the system, the problem of vehicular task allocation in BS is a sequential decision problem, which can be formulated as a Markov decision process (MDP). To solve the problem, we develop a DRL-based vehicular task allocation algorithm to improve the efficiency and the reliability of vehicular computation offloading.

We first consider the VEC scenario illustrated in Fig. 1, where multiple vehicles drive on an one-way road, and several BSs are distributed at the roadside. Similar to [34], we utilize a free flow model to characterize the traffic model in the communication range of a BS, then the relationship between the average velocity of vehicles $\bar{v}$ and traffic density ρ is

$$\bar{v} = v_{max}\left(1 - \frac{\rho}{\rho_{max}}\right), \tag{5}$$

where v_{max} and ρ_{max} are the maximum velocity of vehicles and maximum traffic density, respectively.

4.1 Computation Model

All of the offloading tasks in the task vehicle are assumed to be computation-intensive and are offloaded independently in VEC. The profile of an offloading task is denoted as $\{D_n, C_n, \tau_n\}$, where D_n is the data size of the task, C_n is the computation size which represents the required CPU cycles for completing the task, τ_n is the maximum tolerable delay of the task. In many cases, the execution result of a task is much smaller than data size D_n, we thus ignore the transmission time of execution result in the calculation of offloading delay. Then, the offloading delay of a task is

$$t_n = \frac{D_n}{r_{ts}} + \frac{C_n}{f_n}, \tag{6}$$

where r_{ts} is the transmission rate of the V2V link between task vehicle V_t and service vehicle V_s, f_n is the allocated computing resources for the offloading task in the service vehicle. Similar to [35], we utilize the channel capacity to estimate the V2V transmission rate, which can be given as

$$r_{ts} = W_{ts}\log_2\left(1 + \frac{P_t d_{ts}^{-\alpha} h_{ts}^2}{\sigma_w^2 + I_{ts}}\right), \tag{7}$$

where W_{ts} is the allocated bandwidth of the V2V link between V_t and V_s, P_t is the transmission power of V_t, d_{ts} is the distance between V_t and V_s, α is a constant which represents the path loss factor, h_{ts} is the channel gain of the V2V link, σ_w^2 represents the power spectrum density of additive white Gaussian noise, and I_{ts} denotes the interference introduced by other V2V transmissions.

Particularly, due to the short V2V link duration and limited vehicular computing capability, we do not consider the queuing time in the task offloading, i.e., if a V2V link interruption occurs in the process of task offloading, the task offloading will be regarded as a failure, and the utility of the task will be negative as a penalty. Then, the task vehicle needs to resubmit a new service request to BS for the task.

4.2 Task Vehicle Model

When a task vehicle offloads a task to a service vehicle, it needs to pay the service vehicle the service price, and we assume that the service price of performing a computing task in a service vehicle is proportional to the computation size of the task, which is represented as $p_n C_n$, where p_n denotes the unit price.

For a computing task ϕ_n, similar to [36], [37], the utility is related to the completion time of the task, which is shown as

$$U_n = \begin{cases} \log(1+\tau_n - t_n), & t_n \le \tau_n, \\ -\Lambda, & t_n > \tau_n, \end{cases} \tag{8}$$

where $-\Lambda < 0$ is a constant that represents the penalty of offloading failure. Then, the utility of V_t for offloading a task can be given as

$$\mathcal{U}_n = U_n - p_n C_n. \tag{9}$$

4.3 Service Vehicle Model

In the process of task offloading, once a service vehicle receives a service request and corresponding service price, the service vehicle will contribute part of its computing resources for the offloading task. We consider a service vehicle V_s, and assume that the cost of executing a computing task is proportional to the energy consumption, and denote the computing resources allocated for task ϕ_n as f_n. Similar to [38], the energy consumption of executing task ϕ_n is calculated by

$$E_n = \kappa f_n^3 \frac{C_n}{f_n} = \kappa f_n^2 C_n, \tag{10}$$

where $\kappa > 0$ is a constant. Then, the utility of V_s for executing task ϕ_n can be given as

$$U_n^s = p_n C_n - \varpi \kappa f_n^2 C_n, \tag{11}$$

where ϖ is the coefficient that represents the price per unit of energy consumption.

Since the utility of a service vehicle must be non-negative, the range of f_n is $[0, \min\{F_s, \sqrt{p_n/\kappa}\}]$, where F_s is the maximum available computing resources in V_s, $\sqrt{p_n/\kappa}$ is the upper bound of f_n obtained from (11) with the consideration of the non-negative utility of service vehicle. Moreover, it can be seen from (11) that the utility of a service vehicle mainly depends on the unit service price and the allocated computing resources for task ϕ_n. A service vehicle can increase the utility by allocating fewer computing resources for task ϕ_n, but it will reduce the utility of task ϕ_n or even make the task not completed within the deadline. To prevent some malicious vehicles from allocating insufficient computing resources for offloading tasks, we further establish a reliability model to evaluate the efficiency of task offloading in a service vehicle.

We first define a normalized utility for an offloading task that can be completed within the deadline, which is shown as follows:

$$\tilde{U}_n = \frac{\log(1+\tau_n - t_n)}{\log(1+\tau_n)}, \tag{12}$$

where $\log(1+\tau_n)$ represents the maximum utility of task ϕ_n. After a service vehicle completes an offloading task, the computation efficiency of the service vehicle is updated by

$$\xi_s = (1-\omega_1)\xi_s' + \omega_1 \tilde{U}_n, \quad \omega_1 \in (0,1), \tag{13}$$

where ξ_s' represents the previous computation efficiency of V_s. Besides, the completion ratio of a service vehicle is updated by

$$\zeta_s = \frac{N_s * \zeta_s' + \mathbb{1}(t_n \le \tau_n)}{N_s + 1}, \tag{14}$$

where N_s represents the total number of received offloading tasks in V_s, ζ_s' denotes the previous completion ratio of V_s, and $\mathbb{1}(\cdot)$ is the indicator function. Then, the reliability of a service vehicle can be defined as

$$\eta_s = \omega_2 \xi_s + (1-\omega_2)\zeta_s, \quad \omega_2 \in (0,1). \tag{15}$$

As a result, if a service vehicle cannot allocate appropriate computing resources for an offloading task according to the service price, the reliability of the service vehicle will be reduced. In our proposed V2V computation offloading algorithm, the reliability of service vehicles is an important factor that affects the selection of service vehicles and service price, a vehicle with low reliability usually has a low probability of being selected as the service vehicle for offloading tasks.

4.4 Problem Formulation: Vehicular Task Allocation

In the VEC system, we consider a task vehicle which is denoted as V_t. The service vehicles in the communication range of V_t are denoted as $\mathcal{S} = \{V_1, ..., V_s, ..., V_S\}$. When V_t sends a task offloading request to a BS, the smart contract in the BS is automatically triggered. We assume that there are N offloading tasks from V_t in a period, which are denoted as $\mathcal{N} = \{\phi_1, ..., \phi_n, ..., \phi_N\}$. For each offloading task, the BS dynamically chooses the service vehicle and determines the corresponding service price by observing the vehicular environment. After the task is completed by the service vehicle, the execution result is transmitted to V_t and the BS, and then the BS verifies the result and sends the validity of the result to V_t. If the execution result is valid and the offloading delay is within the maximum tolerable delay of the task, V_t will pay the service price to the service vehicle. Otherwise, the service vehicle will not obtain any payment from V_t. Then, the BS generates a transaction that records the task offloading event. In the V2V computation offloading, we formulate the problem of vehicular task allocation in the BS as an optimization problem with the objective of maximizing the utility of V_t with the consideration of the mobility, the computing capability, and the reliability of vehicles. The optimization problem is formulated as follows:

$$\begin{aligned}\max_{\mathcal{A}} \quad & \frac{1}{N}\sum_{n=1}^{N}\sum_{s=1}^{S} x_n^s \mathcal{U}_n, \\ \text{s.t.} \quad & C1: x_n^s \in \{0,1\}, \quad \forall n \in \mathcal{N}, s \in \mathcal{S}, \\ & C2: \sum_{s=1}^{S} x_n^s = 1, \quad \forall n \in \mathcal{N}, \\ & C3: x_n^s(l_s - t_n) \geq 0, \quad \forall n \in \mathcal{N}, s \in \mathcal{S}, \\ & C4: x_n^s(\eta_s - \eta_0) \geq 0, \quad \forall n \in \mathcal{N}, s \in \mathcal{S}.\end{aligned} \tag{16}$$

where $\mathcal{A} = \{\mathbf{a}_1, ..., \mathbf{a}_n, ..., \mathbf{a}_N\}$, and $\mathbf{a}_n = \{x_n^1, ..., x_n^s, ...x_n^S; p_n\}$. In constraint $C1$, $x_n^s = 1$ means that task ϕ_n is offloaded to V_s, and $x_n^s = 0$ otherwise. Constraint $C2$ guarantees that a computing task is only offloaded to one service vehicle. Constraint $C3$ ensures that the completion time of an offloading task should be less than the V2V link duration between V_t and V_s, and the V2V link duration is estimated by

$$l_s = \frac{\mathcal{D}}{|v_t - v_s|} - \frac{z_t - z_s}{v_t - v_s}, \tag{17}$$

where z_t, v_t represent the position and velocity of task vehicle, respectively, and z_s, v_s represent the position and velocity of service vehicle, respectively. Constraint $C4$ ensures that the reliability of the selected service vehicle is higher than threshold η_0.

In dynamic vehicular networks, the V2V link state, the idle computing resources, and the reliability of service vehicles are all time-variant, and the multi-task allocation can be regarded as a sequential decision problem. We reformulate the optimization problem in (16) as an MDP, and the system state space, action space, and reward are defined as follows.

4.4.1 State Space

The system state at time t can be defined as

$$\begin{aligned}\mathbf{s}_t = [&F_1(t), ..., F_s(t), ..., F_S(t); \gamma_1(t), ..., \gamma_s(t), ..., \gamma_S(t); \\ &l_1(t), ..., l_s(t), ..., l_S(t); \eta_1(t), ..., \eta_s(t), ..., \eta_S(t); \\ &D(t), C(t), \tau(t)],\end{aligned} \tag{18}$$

where $F_s(t)$ represents the available computing resources in V_s, $\gamma_s(t)$ is the signal-to-noise ratio (SNR) of the V2V link between V_t and V_s, $l_s(t)$ denotes the estimated V2V link duration between V_t and V_s, $\eta_s(t)$ is the reliability of V_s, which is estimated by the historical transactions in terms of task offloading in the blockchain according to (15), $D(t)$, $C(t)$, and $\tau(t)$ are the data size, computation size, and deadline of the offloading task at time t, respectively.

4.4.2 Action Space

The conducted action at time t can be given as

$$\mathbf{a}_t = [x_n^1(t), ..., x_n^s(t), ..., x_n^S(t); p_n(t)], \tag{19}$$

where $x_n^s(t) = 1$ represents that service vehicle V_s is selected to execute task ϕ_n, and $x_n^s(t) = 0$ otherwise, and $p_n(t)$ denotes the unit service price of ϕ_n paid to the selected service vehicle.

4.4.3 Reward Function

According to the optimization objective presented in (16), we define the immediate reward after conducting action $\mathbf{a}_t$ at time t as

$$R_t = \sum_{s=1}^{S} x_n^s \mathcal{U}_n, \tag{20}$$

where $\mathcal{U}_n$ is the utility of V_t for offloading a task as defined in (9), which is related to service vehicle and service price.

4.5 SAC-Based Computation Offloading Algorithm

In the problem of vehicular task allocation, the action in terms of the selection of service vehicles and the determination of service price contains both discrete decision variables and continuous variable, we thus employ soft actor-critic (SAC) [39] method to obtain the hybrid decision policy in vehicular task allocation. SAC is a reinforcement learning algorithm which is based on off-policy actor-critic model. To be specific, the actor network in SAC is used to generate action according to the observed system state and improve the policy, while the critic network is responsible for evaluating the policy provided by the actor network. In the following, we will demonstrate how SAC works in task allocation.

4.5.1 Policy and Value Function

The main goal of SAC is to find an optimal policy that maximizes the expected long-term reward, and meanwhile maximizes the entropy of policy to improve the robustness and exploration. The policy is defined as an action-selection strategy which can be either deterministic or stochastic. In SAC, the policy is stochastic and is denoted as $\pi(\mathbf{a}|\mathbf{s})$, which represents a probability distribution over actions given an observed state $\mathbf{s}$. We then define a soft action-value function that represents the expected long-term reward combined with the expected entropy of policy π when given an initial state $\mathbf{s}$ and action $\mathbf{a}$, which is shown as

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nu^t R_t + \beta \sum_{t=1}^{T-1} \nu^t H(\pi(\cdot|\mathbf{s}_t)) | \mathbf{s}_0 = \mathbf{s}, \mathbf{a}_0 = \mathbf{a}\right], \tag{21}$$

where $\beta \in [0, 1]$ is the temperature parameter that controls the stochasticity of the policy, and ν is the discount factor. Further, we define a soft state-value function that denotes the expected long-term reward combined with the entropy of policy starting from state $\mathbf{s}$ and taking actions following some policy π, which is given as follows:

$$V^{\pi}(\mathbf{s}) = \mathbb{E}\left[\sum_{t=0}^{T-1} \nu^t \left(R_t + \beta H\left(\pi(\cdot|\mathbf{s}_t)\right)\right) | \mathbf{s}_0 = \mathbf{s}\right]. \tag{22}$$

Since the dimensions of both state space and action space could be extremely large with the increase of traffic density, we employ neural networks to approximate both the soft action-value function and the policy, which are denoted as $Q_\theta(\mathbf{s}, \mathbf{a})$ and $\pi_\psi(\mathbf{a}|\mathbf{s})$, respectively.

4.5.2 Critic Network Training

In our algorithm, the critic network is in charge of policy evaluation. In each step, the agent of BS samples a batch of experiences from replay buffer $\mathcal{B}$, and then trains the critic network. Specifically, the policy is measured by the mean square error (MSE) between the soft action-value function and the target soft action-value function, and the loss function is given as follows:

$$L_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t)\sim\mathcal{B}}\left[\frac{1}{2}\left(Q_\theta(\mathbf{s}_t,\mathbf{a}_t) - \hat{Q}_{\hat{\theta}}(\mathbf{s}_t,\mathbf{a}_t)\right)^2\right], \quad (23)$$

where the target soft action value function is defined as

$$\hat{Q}_{\hat{\theta}}(\mathbf{s}_t,\mathbf{a}_t) = R_t + \nu\mathbb{E}_{\mathbf{s}_{t+1}\sim\rho_\pi}\left[V_{\hat{\theta}}(\mathbf{s}_{t+1})\right]. \quad (24)$$

Then, the critic network is trained by minimizing the loss function in (23) in each step.

4.5.3 Actor Network Training

The actor network is in charge of generating actions by the observed states. According to [39], the actor network is trained by minimizing the Kullback-Leibler (KL) divergence, which is presented as follows:

$$\pi_t = \arg\min_{\pi'\in\Pi} \mathrm{D}_{\mathrm{KL}}\left(\pi'(\cdot|\mathbf{s}_t) \parallel \frac{\exp(\frac{1}{\beta}Q^\pi(\mathbf{s}_t,\cdot))}{Z^\pi(\mathbf{s}_t)}\right), \quad (25)$$

where $Z^\pi(\mathbf{s}_t)$ is the function that normalizes the distribution, which does not affect the gradient of the policy. We transform the expression of π_t in (25) by multiplying β and ignoring the term $Z^\pi(\mathbf{s}_t)$, and then obtain the loss function of policy, which is given as follows:

$$L_\pi(\psi) = \mathbb{E}_{\mathbf{s}_t\sim\mathcal{B}}\left[\mathbb{E}_{\mathbf{a}_t\sim\pi_\psi}\left[\beta\log\pi_\psi(\mathbf{a}_t|\mathbf{s}_t) - Q_\theta(\mathbf{s}_t,\mathbf{a}_t)\right]\right]. \quad (26)$$

In each step, the actor network is trained by minimizing the loss function in (26).

4.5.4 Temperature Adjustment

In the above descriptions of the training process of actor and critic networks, temperature parameter β is regarded as a constant. According to [40], the temperature parameter can be learned in each step by minimizing the loss function

$$L_t(\beta) = \mathbb{E}_{\mathbf{a}_t\sim\pi}\left[-\beta\log\pi(\mathbf{s}_t,\mathbf{a}_t) - \beta H_0\right], \quad (27)$$

where H_0 is the entropy threshold.

Moreover, to mitigate the positive bias in policy improvement, two critic networks are established in SAC [40], and both of which are trained independently. Then in the calculation of the stochastic gradient of $L_Q(\theta)$ and the policy gradient of $\pi(\mathbf{a}|\mathbf{s})$, we use the minimum soft Q-value of the two critic networks.

Finally, the detailed vehicular task allocation algorithm is presented in Algorithm 1.

Algorithm 1 SAC-Based Task Allocation Algorithm

```
Initialize: Initialize critic networks $Q_{\theta_1}(\mathbf{s},\mathbf{a})$ and $Q_{\theta_2}(\mathbf{s},\mathbf{a})$ with weights $\theta_1$ and $\theta_2$, respectively.
Initialize target critic networks $\hat{Q}_{\hat{\theta}_1}(\mathbf{s},\mathbf{a})$ and $\hat{Q}_{\hat{\theta}_2}(\mathbf{s},\mathbf{a})$ with weights $\hat{\theta}_1$ and $\hat{\theta}_2$, respectively.
Initialize actor network $\pi_\psi(\mathbf{a}|\mathbf{s})$ with weights $\psi$.
Initialize replay buffer $\mathcal{B} = \emptyset$.
for each period do
  for each task $\phi_n$ do
    The BS collects information of vehicles and evaluates current state $\mathbf{s}_t$.
    Generates action $\mathbf{a}_t$ that determines service vehicle $V_s$ and unit price $p_n$.
    Sends a task allocation message to $V_t$ and $V_s$.
    $V_s$ allocates computing resources for task $\phi_n$.
    The BS calculates the immediate reward $R_t$ and estimates the next state $\mathbf{s}_{t+1}$.
    Store $(\mathbf{s}_t,\mathbf{a}_t,R_t,\mathbf{s}_{t+1})$ into replay buffer $\mathcal{B}$.
  end for
  Sample a batch of experiences $\mathcal{E}$ from $\mathcal{B}$.
  Update weights of $Q_{\theta_1}(\mathbf{s},\mathbf{a})$ and $Q_{\theta_2}(\mathbf{s},\mathbf{a})$ by computing the gradient of $L_Q(\theta_i)$ in (23),
      $\theta_i = \theta_i - \lambda_Q\nabla_{\theta_i}\frac{1}{\mathcal{E}}\sum_{\mathcal{E}} L_Q(\theta_i), \quad i = 1,2.$
  Update weights of $\pi_\psi(\mathbf{a}|\mathbf{s})$ by computing the gradient of $L_\pi(\psi)$ in (26),
      $\psi = \psi - \lambda_\pi\nabla_\psi\frac{1}{\mathcal{E}}\sum_{\mathcal{E}} L_\pi(\psi).$
  Update $\beta$ by computing the gradient of $L_t(\beta)$ in (27),
      $\beta = \beta - \lambda_\beta\nabla_\beta\frac{1}{\mathcal{E}}\sum_{\mathcal{E}} L_t(\beta).$
  Update weights of $\hat{Q}_{\hat{\theta}_1}(\mathbf{s},\mathbf{a})$ and $\hat{Q}_{\hat{\theta}_2}(\mathbf{s},\mathbf{a})$ by
      $\hat{\theta}_i = \omega_s\hat{\theta}_i + (1-\omega_s)\theta_i, \quad i = 1,2.$
end for
```

5 DRL-Based consensus nodes selection

In our proposed V2V computation offloading scheme, the reliability of service vehicles in resource allocation is a main factor that affects the decision in vehicular task allocation. In the blockchain-enabled VEC, the reliability of a vehicle in resource allocation is evaluated by the transactions of vehicular computation offloading, and the accuracy of the estimated reliability highly depends on the timeliness of the transactions. Therefore, it is necessary to design a consensus scheme to optimize the efficiency of transaction verification.

In this section, we propose a consensus nodes selection algorithm for the PBFT-based consensus in the blockchain-enabled VEC system, where the consensus nodes and the block size in each round of blockchain consensus are both determined according to the states of BSs. To maximize the efficiency of blockchain consensus and meanwhile motivate BSs to improve their reliability in vehicular task allocation, we model the consensus nodes and incorporate the incentive mechanism into the reward of consensus nodes in Section 5.1. Besides, considering that the available computing resources in BSs for consensus and the reliability of BSs are both time-variant, we employ a DRL-based algorithm in Section 5.3 so that the policy of consensus nodes selection can dynamically adapt to the state of the VEC system.

5.1 Consensus Node Model

In the blockchain-enabled VEC system, we assume that there are B BSs distributed in a certain area, which are denoted as $\mathcal{I} = \{I_1, ..., I_b, ..., I_B\}$. In each round of blockchain consensus, all the consensus nodes are selected by the primary node of the previous round of consensus. According to the analysis of blockchain consensus presented in Section 3.4, the performance of consensus mainly depends on the block size and available computing resources in BSs. Considering that with the traffic density in the communication range of a BS increasing, there will be more vehicles sending computing service requests to the BS, and without loss of generality, we assume that the arrival rate of vehicular service requests in a BS is proportional to the traffic density in the communication range of the BS. Then, the available computing resources for blockchain consensus in the BS can be given as

$$G_b = G_b^{total} - (\tilde{G}_b + \mu\rho_b), \tag{28}$$

where G_b^{total} is the total computing capability of BS I_b, $\tilde{G}_b$ represents the computing resources reserved for some non-vehicular computing tasks in I_b, ρ_b is the traffic density in the communication range of I_b, and $\mu > 0$ is a constant. Following similar assumption of block reward in [38], we define the reward of a consensus node after completing a round of consensus as

$$R_b(t) = \begin{cases} \left(\varepsilon_1 \iota_b + \varepsilon_2 \frac{S_{bk}}{\chi}\right) \log(1 + \mathcal{T} - T_b^V), & T_b^V \leq \mathcal{T}, \\ -\Gamma, \quad T_b^V > \mathcal{T}, \end{cases} \tag{29}$$

where ι_b is the reliability of BS I_b, ε_1 and ε_2 are the weights of reliability and number of validated transactions, respectively, T_b^V denotes the delay of block verification in I_b, $\mathcal{T} = \delta T^I - (T^I + T^D)$ represents the maximum tolerable delay of block verification in I_b, and $-\Gamma < 0$ is a constant that represents the penalty. It can be seen from (29) that the reward of a consensus node is simultaneously determined by the block size and the reliability, the BS that validates more transactions and has higher reliability will get more reward in the consensus process. Therefore, BSs are motivated to improve their reliability in vehicular task allocation and contribute more computing resources for blockchain consensus. After a round of consensus is completed, the average reward obtained by the consensus nodes from blockchain is

$$\mathcal{R}(t) = \frac{1}{N_c} \sum_{b=1}^{N_c} R_b(t). \tag{30}$$

5.2 Problem Formulation: consensus nodes selection

Our aim is to maximize the expected long-term reward of consensus nodes. According to the reward of consensus nodes defined in (30), we formulate the optimization problem as

$$\begin{aligned} \max_{\mathcal{A}'} \quad & \sum_{t'=t}^{T} \epsilon^{t'-t} \mathcal{R}(t'), \\ \text{s.t.} \quad & C1: y_b(t') \in \{0, 1\}, \quad \forall b \in \mathcal{I}, \\ & C2: \sum_{b=1}^{B} y_b(t') = N_c, \\ & C3: y_b(t')(G_b - \mathcal{G}) \geq 0, \quad \forall b \in \mathcal{I} \\ & C4: y_b(t')(\iota_b(t') - \mathcal{L}) \geq 0, \quad \forall b \in \mathcal{I} \\ & C5: T^F - \delta T^I \leq 0, \end{aligned} \tag{31}$$

where $\mathcal{A}' = \{y_1(t'), ..., y_b(t'), ..., y_B(t'); S_{bk}(t')\}$, and $\epsilon \in (0, 1]$ is the reward discount factor. In constraint $C1$, $y_b(t') = 1$ indicates that BS I_b is selected as a consensus node at time t', and $y_b(t') = 0$ otherwise. Constraint $C2$ indicates that the total number of consensus nodes is N_c. Constraint $C3$ guarantees that the available computing resources in a consensus node should be higher than threshold $\mathcal{G}$. Constraint $C4$ ensures that the reliability of a consensus node should be higher than threshold $\mathcal{L}$. In constraint $C5$, the consensus delay should satisfy the constraint in (4).

In the consensus nodes selection, the available computing resources for blockchain consensus in each BS depends on the traffic density in the communication range of the BS, the reliability of each BS depends on the performance in vehicular task allocation, and both of which are time-variant. Moreover, the consensus nodes selection is a joint decision-making problem, where the block size and multiple consensus nodes are jointly determined in each round of consensus, which is difficult to be solved by traditional optimization methods. In the following, we will transform the problem as an MDP and demonstrate how to solve the problem by using a DRL-based method.

5.2.1 *State Space*

In each period, the system state of all the BSs is denoted as

$$\begin{aligned} \mathbf{s}_t = [&G_1(t), ..., G_b(t), ..., G_B(t); \\ &\iota_1(t), ..., \iota_b(t), ..., \iota_B(t); N_{tr}(t)], \end{aligned} \tag{32}$$

where $G_b(t)$ is the available computing resources for blockchain consensus in BS I_b at time t, $N_{tr}(t)$ represents the number of transactions generated by BSs in a block interval, $\iota_b(t)$ represents the reliability of I_b in vehicular task allocation, which is evaluated by the average normalized utility defined in (12) and the completion ratio of the vehicular offloading tasks allocated by the BS, and the calculation of $\iota_b(t)$ is the same as (15). It is worth noting that the reliability of a BS reflects the task allocation in the BS, which depends on the selection of service vehicle and service price, while the reliability of a service vehicle reflects the task execution in the service vehicle, which depends on the number of allocated computing resources allocated for offloading tasks.

5.2.2 *Action Space*

We represent the action conducted by the agent at time t as

$$\mathbf{a}_t = [y_1(t), ..., y_b(t), ..., y_B(t); S_{bk}(t)], \tag{33}$$

where $y_b(t) = 1$ indicates that BS I_b is selected as a consensus node, and $y_b(t) = 0$ otherwise. $S_{bk}(t)$ denotes the size of unverified block, and $S_{bk}(t) \in \{0.2, 0.4, ..., \dot{S}_{bk}\}$, where $\dot{S}_{bk}$ is the maximum block size in the blockchain. Furthermore, each consensus node has the potential to be selected as the primary node, and the other consensus nodes are the replica nodes. Among the consensus nodes chosen from $\mathbf{a}_t$, we select the consensus node that maximizes $\mathcal{R}(t)$ as the primary node.

5.2.3 Reward Function

According to the optimization objective defined in (31), the immediate reward in a round of consensus can be given as

$$R(\mathbf{s}_t, \mathbf{a}_t) = \mathcal{R}(t). \tag{34}$$

5.3 consensus nodes selection Algorithm

In the problem of consensus nodes selection, the selection of consensus nodes are discrete decision variables, and the block size is determined by the number of transactions in a block, which is also a discrete decision variable. Therefore, the action space is discrete. We thus employ double deep Q-network (DDQN) [41], which is suitable to solve discrete action problems and can guarantee global optima. The aim of our algorithm is to find an optimal policy that maximizes the expected long-term reward of the consensus nodes, we first define a Q-value function $Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t)$ to represent the expected long-term reward after performing action $\mathbf{a}_t$ at state $\mathbf{s}_t$ according to some policy π, which is shown as

$$Q^{\pi}(\mathbf{s}_t, \mathbf{a}_t) = \mathbb{E}\left[R_t + \nu' Q^{\pi}(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}) | \mathbf{s}_t, \mathbf{a}_t\right]. \tag{35}$$

Then, the optimal policy that maximizes the expected long-term reward is represented as

$$\pi(\mathbf{s}_t) = \arg\max_{\mathbf{a}} Q(\mathbf{s}_t, \mathbf{a}). \tag{36}$$

To obtain the optimal policy, a value iteration of the Q-value function is employed, which can be given as follows:

$$Q(\mathbf{s}_t,\mathbf{a}_t) = Q(\mathbf{s}_t,\mathbf{a}_t) + \beta'\left(R_{t+1} + \nu' \max_{\mathbf{a}} Q(\mathbf{s}_{t+1},\mathbf{a}) - Q(\mathbf{s}_t,\mathbf{a}_t)\right). \tag{37}$$

Considering that the dimensions of state space and action space could be extremely large, it is difficult to obtain the optimal policy by looking up the table that stores the Q-value of the historical experiences. Therefore, in DDQN, the neural network is employed to approximate the Q-value function, which is called Q-network and denoted as $Q(\mathbf{s}, \mathbf{a}; \theta)$, where θ represents the weights of Q-network. Then, the optimal policy becomes $\pi(\mathbf{s}_t) = \arg\max_{\mathbf{a}} Q(\mathbf{s}_t, \mathbf{a}; \theta)$.

To mitigate the overestimation in the learning process, two Q-networks are established and trained independently, i.e., the main Q-network and the target Q-network, which are represented as $Q(\mathbf{s}, \mathbf{a}; \theta)$ and $Q^-(\mathbf{s}, \mathbf{a}; \theta^-)$, respectively. The main Q-network is responsible for choosing action according to the observed system state, while the target Q-network is used to evaluate the policy obtained from the main Q-network. In the target Q-network, the target Q-value can be calculated as follows:

$$z_t^- = R_t + \nu' Q^-(\mathbf{s}_{t+1}, \arg\max_{\mathbf{a}} Q(\mathbf{s}_{t+1}, \mathbf{a}; \theta); \theta^-). \tag{38}$$

Since we aim to make the Q-value of the main Q-network close to the target Q-value, the loss function of the main Q-network is defined as follows:

$$J(\theta) = \frac{1}{\mathcal{E}'} \sum_{i=1}^{\mathcal{E}'} \left(z_i^- - Q(\mathbf{s}_i, \mathbf{a}_i; \theta)\right)^2. \tag{39}$$

In each period, the agent samples a batch of experiences $\mathcal{E}'$ from the replay buffer, and trains the main Q-network by minimizing the loss function in (39). Finally, after every N^- steps, the weights of the target Q-network θ^- is updated with an exponential moving average of θ. The detailed algorithm is presented in Algorithm 2.

Algorithm 2 consensus nodes selection Algorithm

```
Initialize: Initialize main Q-network $Q(\mathbf{s}, \mathbf{a}; \theta)$ and target Q-network $Q^-(\mathbf{s}, \mathbf{a}; \theta^-)$ with random weights $\theta$ and $\theta^-$, respectively.
Initialize replay buffer $\mathcal{B}' = \emptyset$.
for each period do
    Primary node evaluates current state $\mathbf{s}_t$ according to the computing resources and reliability of each BS.
    Generate a random probability $p$.
    if $p < \epsilon$, then
        Randomly select action $\mathbf{a}_t$ from the action space.
    else
        Choose action $\mathbf{a}_t = \arg\max_{\mathbf{a}} Q(\mathbf{s}_t, \mathbf{a}; \theta)$.
    end if
    Broadcast the message that contains the consensus nodes selection and block size.
    Store $(\mathbf{s}_t, \mathbf{a}_t, R_t, \mathbf{s}_{t+1})$ into replay buffer $\mathcal{B}'$.
    Calculate target Q-value $z_t^-$ by (38).
    Update the weights of $Q(\mathbf{s}, \mathbf{a}; \theta)$ by computing the gradient of $J(\theta)$ in (39), $\theta = \theta - \lambda' \nabla_\theta J(\theta)$.
    Update the weights of $Q^-(\mathbf{s}, \mathbf{a}; \theta^-)$ every $N^-$ steps by $\theta^- = \omega_d \theta + (1 - \omega_d)\theta^-$.
end for
```

6 Performance Evaluation

6.1 Simulation Setup

We conduct our simulation on a desktop, which has two NVIDIA TITAN Xp GPUs, a 128G RAM, and an Intel Xeon CPU. The simulation platform is based on Pytorch with Python 3.7 on Ubuntu 16.04 LTS.

In the simulation of V2V computation offloading, we consider a one-way road, where multiple vehicles drive in the same direction, and a BS deployed by the road takes charge of vehicular task allocation. We set the number of offloading tasks of a task vehicle in a period as 32, and simulate the performance of our proposed algorithm under different traffic densities in the communication range of the BS. Besides, in the simulation of blockchain consensus, we consider an area with 50 BSs in total, and the traffic density in the communication range of the BSs varies from 5 vehicles/km to 40 vehicles/km. We conduct multiple simulations with different numbers of consensus nodes N_c, and the range of N_c is set as $[6, 30]$. The other parameters in our simulation are summarized in Table 2.

TABLE 2
Simulation Parameters

Parameter	Value
ρ	$5 \sim 40$ vehicles/km
F_s	$[3, 7]$ GHz
G_b^{total}	$[20, 29]$ GHz
$\mathcal{D}$	500 m
W_{ts}	10 MHz
D_n	$[0.2, 4]$ Mbits
C_n	$[0.2, 3.2] \times 10^9$ cycles
τ_n	$\{0.5, 1, 2, 4\}$ s
χ	2 KB
$\dot{S}_{bk}$	8 MB [42]
a	2 Mcycles
b	8 Mcycles [31]
c	0.5 Mcycles [31]

6.2 Performance of V2V Computation Offloading

In the simulation of the vehicular computation offloading algorithm, actor network $\pi_\psi(\mathbf{a}|\mathbf{s})$ and critic networks $Q_{\theta_1}(\mathbf{s}, \mathbf{a})$ and $Q_{\theta_2}(\mathbf{s}, \mathbf{a})$ are all fully-connected deep neural networks (DNNs) with two hidden layers. We set the size of the hidden layers in $\pi_\psi(\mathbf{a}|\mathbf{s})$ as $(800, 500)$, and set the learning rate of $\pi_\psi(\mathbf{a}|\mathbf{s})$ as $\lambda_\pi = 8 \times 10^{-4}$. Similarly, we set the size of the hidden layers in both $Q_{\theta_1}(\mathbf{s}, \mathbf{a})$ and $Q_{\theta_2}(\mathbf{s}, \mathbf{a})$ as $(800, 500)$, and the learning rate of the two networks is set as $\lambda_Q = 8 \times 10^{-4}$. In addition, the batch size is set as $|\mathcal{E}| = 256$, and the delay factor is set as $\omega_s = 0.005$.

The complexity of the vehicular computation offloading algorithm is commensurate with that of the DNN model. Given a VEC scenario where the number of service vehicles is S, then the dimension of the state space is $4S + 3$ and the dimension of the action space is $2S$ according to (18) and (19). Then, the complexity of action generation in the algorithm can be given as $O((4S + 3)h_1 + h_1 h_2 + 2h_2 S)$, where $(h1, h2)$ is the size of the hidden layers in both actor network and critic network. In our simulations, the size of hidden layers in actor and critic is fixed, the complexity of action generation can be given as $O(S)$. Besides, in the training process of the algorithm, the complexity in each step is $O(S)$ as well. If there are N_o offloading tasks in a period, and the training steps in a period are N_t, then the time complexity of the algorithm in a period is $O((N_o + N_t)S)$.

To verify the performance of our proposed algorithm, we introduce the following algorithms for comparison:

- Greedy task offloading (GTO): For each offloading task, the BS always chooses the service vehicle with the maximum idle computing resources, and gives the service price that maximizes the utility of the task.
- Random task offloading (RTO): For each offloading task, the BS randomly chooses a service vehicle, and randomly gives a service price in the price domain.

Fig. 4 shows the average utility of a task vehicle for offloading a task in our proposed algorithm with different learning rates. It can be seen that the average utility reaches convergence around 3,000 training episodes, and the algorithm with learning rates $\lambda_Q = 8 \times 10^{-4}, \lambda_\pi = 8 \times 10^{-4}$ achieves the best performance compared with the algorithm with other learning rates.

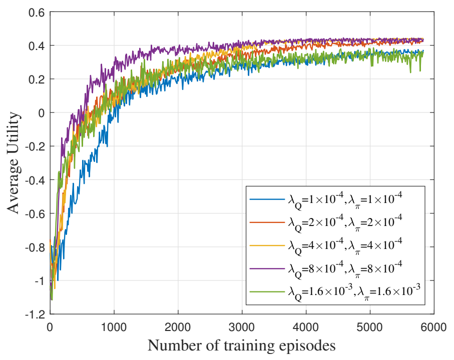


Fig. 4. The average utility of task vehicle for offloading a task in the proposed algorithm with different learning rates.

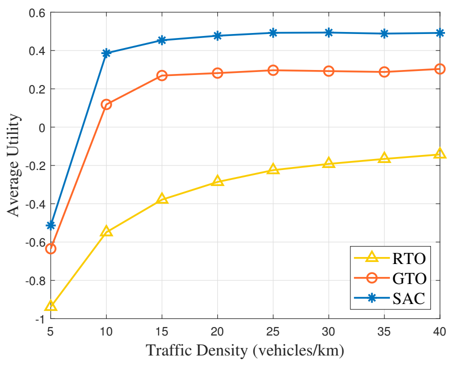


Fig. 5. The average utility of task vehicle for offloading a task under different traffic densities.

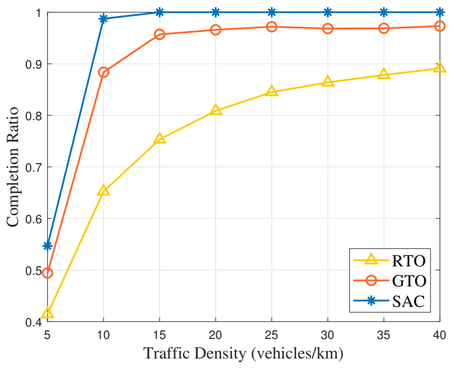
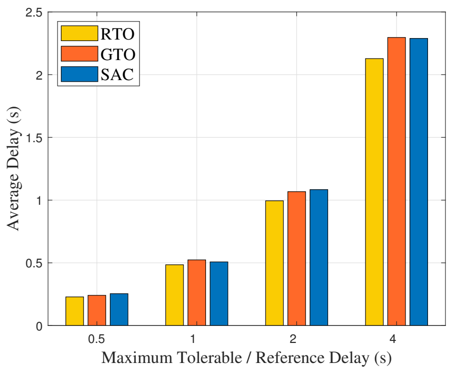


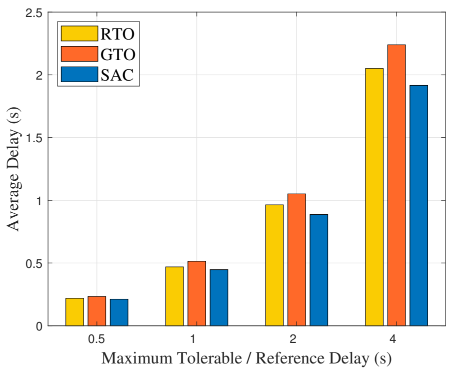
Fig. 6. The completion ratio of offloading tasks under different traffic densities.

As shown in Fig. 5, the average utility of task vehicle for offloading a task in our proposed algorithm is higher than that in GTO and RTO, because the optimization objective of our proposed algorithm is to maximize the expected long-term reward, and the reward is related to the utility of task vehicle, while in GTO, the agent always chooses the service vehicle with the maximum idle computing resources in each step, which may not obtain the maximum average utility of offloading tasks. Moreover, GTO does not consider the reliability of service vehicles in task offloading, some tasks may be offloaded to service vehicles with low reliability and obtain fewer computing resources, which leads to the lower average utility of the task vehicle.

In Fig. 6, the completion ratio of offloading tasks by



(a) The average delay of offloading tasks when the traffic density is 10 vehicles/km.



(b) The average delay of offloading tasks when the traffic density is 30 vehicles/km.

Fig. 7. The average delay of offloading tasks with different maximum tolerable delays.

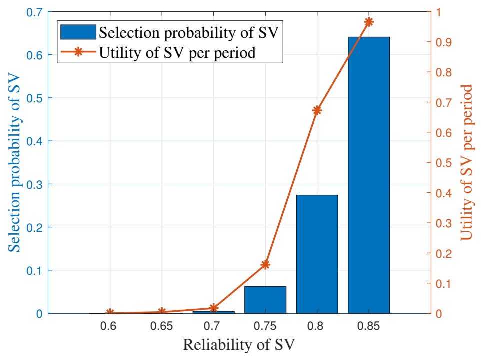


Fig. 8. The average utility per period and the selection probability of service vehicles (SVs) with different reliabilities when the traffic density is 30 vehicles/km.

applying the three algorithms is presented. It can be seen that the completion ratio of offloading tasks by applying our proposed algorithm is higher than by applying GTO and RTO. Furthermore, with the traffic density increasing, there are more service vehicles available for task offloading, thus offloading tasks may obtain more computing resources from service vehicles, which increases both the average utility and the completion ratio of offloading tasks.

Fig. 7(a) and Fig. 7(b) present the average delays of the offloading tasks that have been successfully completed under the traffic density of 10 vehicles/km and 30 vehicles/km, respectively. We can observe that under low traffic density, the average delays of offloading tasks with $\tau_n \in \{0.5, 1, 2, 4\}$ in our algorithm are close to that in GTO, but the completion ratio of GTO is lower than that of our proposed algorithm as illustrated in Fig. 6. Moreover, under high traffic density, it can be seen that the average delays of offloading tasks with different deadlines by applying our proposed algorithm are all lower than the other two algorithms. This is because our proposed algorithm considers the reliability of service vehicles and meanwhile maximizes the expected long-term reward that is related to the offloading delay, while the other two algorithms do not consider these aspects in vehicular task allocation.

In addition, Fig. 8 presents the probability of being selected as a service vehicle for an offloading task and the average utility per period of service vehicles with different reliabilities by applying our proposed algorithm. It can be seen that vehicles with higher reliability are more likely to be chosen as the service vehicles for offloading tasks. Meanwhile, service vehicles with higher reliability can obtain higher utility in a period, this is because service vehicles with higher reliability can receive more offloading tasks in vehicular task offloading.

6.3 Performance of Consensus Process

In the simulation of the consensus nodes selection algorithm, we utilize fully-connected DNNs to represent the Q-networks. Both the main Q-network $Q(\mathbf{s}, \mathbf{a}; \theta)$ and target Q-network $Q^-(\mathbf{s}, \mathbf{a}; \theta^-)$ have two hidden layers with size $(1000, 1000)$, and the learning rate of the algorithm is set as $\lambda' = 5 \times 10^{-5}$. Besides, we set the batch size $|\mathcal{E}'| = 256$, and set the delay factor $\omega_d = 0.01$.

The complexity of the consensus nodes selection algorithm is commensurate with that of the DNN model as well. Similar to the complexity analysis of Algorithm 1, the complexity of action generation and training in each step is $O(N_B)$, where N_B is the number of BSs in the system.

To validate the performance of the DRL-based algorithm of consensus nodes selection, we provide three benchmark algorithms for comparison:

- traditional PBFT (PBFT): In each round of consensus, the consensus nodes are preassigned and do not vary with the rounds of consensus. The primary node in each round of consensus is selected in a round-robin manner, and the block size is selected to maximize the average reward of the consensus nodes.
- Greedy node selection (GNS): In each round of consensus, GNS always selects N_c BSs with the most available computing resources as the consensus nodes, and chooses the block size that maximizes the average reward of the consensus nodes.
- Random node selection (RNS): In each round of consensus, RNS selects N_c BSs in a random manner, and chooses the block size that maximizes the average reward of the consensus nodes.

Fig. 9 shows the average utility of consensus nodes in a round of consensus in our proposed algorithm with different learning rates. We can notice that our proposed algorithm with learning rate $\lambda' = 5 \times 10^{-5}$ or $\lambda' = 1 \times 10^{-4}$

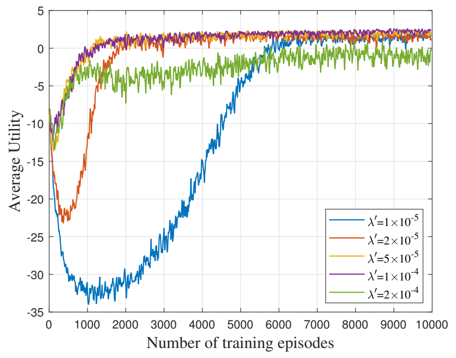


Fig. 9. The average utility of consensus nodes in a round of consensus in the proposed algorithm with different learning rates.

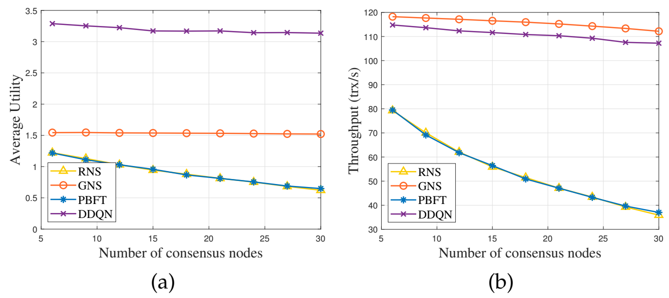


Fig. 10. The average utility and throughput of consensus nodes in a round of consensus with different number of consensus nodes.

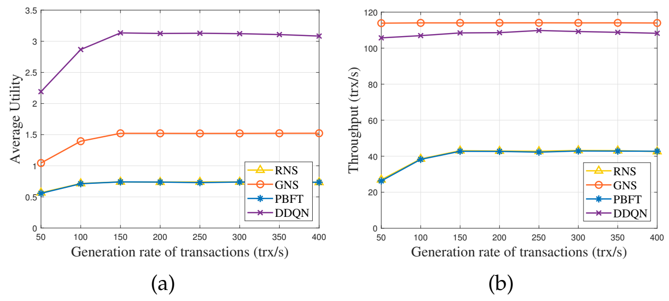


Fig. 11. The average utility and throughput of blockchain consensus under different generation rates of transactions.

reaches convergence around 2,000 training episodes. Besides, it can be seen from Fig. 9 that when the learning rate becomes smaller, the proposed algorithm needs to take more episodes to achieve convergence. When the learning rate becomes large enough, the proposed algorithm falls into a local optimum and cannot achieve the best performance.

As shown in Fig. 10(a), the average utility of consensus nodes in our proposed algorithm is higher than that in the benchmark algorithms, because our proposed algorithm considers both the available computing resources and the reliability of BSs, and jointly optimizes the consensus nodes selection and block size, while benchmark algorithms do not consider the reliability of the BSs in the selection of consensus nodes. Additionally, we can notice that the performance of RNS is close to that of PBFT, this is because both RNS and PBFT do not utilize the state information of BSs in the consensus nodes selection.

Fig. 10(b) presents the throughput of blockchain consensus with different numbers of consensus nodes by applying our proposed algorithm and benchmark algorithms. We can notice that GNS achieves higher throughput than the other algorithms, because GNS chooses the BSs with the maximum available computing resources and does not consider the reliability of BSs, which makes the block verification completed in the least time. Besides, we can notice that the throughput of consensus nodes decreases with the number of consensus nodes increasing, this is because with the increase of consensus nodes, the BSs with fewer available computing resources will have more opportunities to participate in the consensus, and the time of block verification in a round of consensus depends on the minimum available computing resources of the consensus nodes.

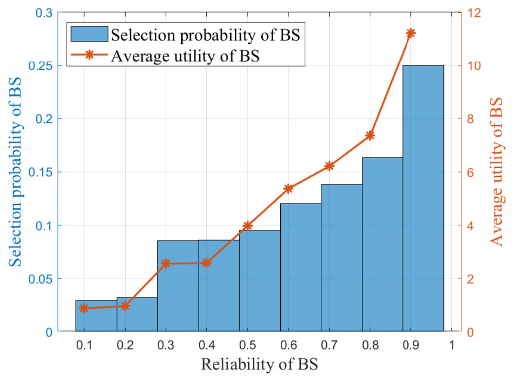


Fig. 12. The average utility and selection probability of BSs with different reliabilities in blockchain consensus.

Fig. 11(a) and Fig. 11(b) show the average utility and the throughput of blockchain consensus under different generation rates of transactions. From Fig. 11(a), it can be seen that the average utility of consensus nodes in our proposed algorithm is higher than that in the other algorithms under different generation rates of transactions. Moreover, since the computing resources in BSs are limited, with the generation rates of transactions increasing, the average utility of consensus nodes gradually becomes larger and then tends to be stable. From Fig. 11(b), we can notice that with the generation rates of transactions increasing, the throughput of blockchain consensus gradually increases and then tends to be stable in our proposed algorithm, while the throughput of blockchain consensus in GNS remains unchanged, this is because GNS always chooses the BSs with the most computing resources as the consensus nodes, the throughput of GNS represents the upper bound of the throughput in consensus.

In addition, Fig. 12 shows that the BS with higher reliability has a higher probability of being selected as a consensus node in our proposed algorithm. Meanwhile, BS with higher reliability can receive more rewards from blockchain by participating in the consensus process. As presented in (29), the utility of a consensus node not only depends on the consensus delay and the number of validated transactions, but also depends on the reliability of the BS. Since the objective of our proposed algorithm is to maximize the average utility of consensus nodes, the agent tends to select BSs with high reliability as the consensus nodes. Besides, through the reward from blockchain, BSs are motivated to improve their reliability in vehicular task allocation.

7 Conclusion

In this paper, we have investigated the integration of blockchain and VEC, and proposed a vehicular task allocation scheme to ensure secure and reliable computation offloading among vehicles in the smart contract. To make the policy of computation offloading adapt to the dynamic vehicular environment, we formulated the problem of vehicular task allocation as a sequential decision problem, and developed a DRL-based algorithm to solve the problem. Furthermore, to improve the efficiency of blockchain consensus and motivate BSs to improve the reliability in vehicular task allocation, we proposed an enhanced consensus protocol based on PBFT, and then designed a DDQN-based algorithm for consensus nodes selection. Finally, simulation results revealed that the proposed algorithm can effectively improve the performance of V2V computation offloading and blockchain consensus, and meanwhile motivate BSs to improve their reliability in vehicular resource allocation.

References

[1] X. Wang, Z. Ning, S. Guo, and L. Wang, "Imitation learning enabled task scheduling for online vehicular edge computing," *IEEE Trans. Mobile Comput.*, to be published, doi:10.1109/TMC.2020.3012509.

[2] J. Du, C. Jiang, J. Wang, Y. Ren, and M. Debbah, "Machine learning for 6G wireless networks: Carry-forward-enhanced bandwidth, massive access, and ultrareliable/low latency," *IEEE Veh. Technol. Mag*, vol. 15, no. 4, pp. 123–134, Dec. 2020.

[3] X. Peng, K. Ota, and M. Dong, "Multiattribute-based double auction toward resource allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3094–3103, Apr. 2020.

[4] Z. Ning, P. Dong, X. Wang, X. Hu, J. Liu, L. Guo, B. Hu, R. Kwok, and V. C. M. Leung, "Partial computation offloading and adaptive task scheduling for 5G-enabled vehicular networks," *IEEE Trans. Mobile Comput.*, to be published, doi:10.1109/TMC.2020.3025116.

[5] Z. Gao, M. Liwang, S. Hosseinalipour, H. Dai, and X. Wang, "A truthful auction for graph job allocation in vehicular cloud-assisted networks," *IEEE Trans. Mobile Comput.*, to be published, doi:10.1109/TMC.2021.3059803.

[6] J. Du, C. Jiang, H. Zhang, Y. Ren, and M. Guizani, "Auction design and analysis for SDN-based traffic offloading in hybrid satellite-terrestrial networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2202–2217, Oct. 2018.

[7] H. Xu, W. Huang, Y. Zhou, D. Yang, M. Li, and Z. Han, "Edge computing resource allocation for unmanned aerial vehicle assisted mobile network with blockchain applications," *IEEE Trans. Wireless Commun.*, vol. 20, no. 5, pp. 3107–3121, May 2021.

[8] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 33–39, Aug. 2018.

[9] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4585–4600, Jun. 2019.

[10] M. B. Mollah, J. Zhao, D. Niyato, Y. L. Guan, C. Yuen, S. Sun, K. Y. Lam, and L. H. Koh, "Blockchain for the internet of vehicles towards intelligent transportation systems: A survey," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4157–4185, Mar. 2021.

[11] J. Feng, F. R. Yu, Q. Pei, J. Du, and L. Zhu, "Joint optimization of radio and computational resources allocation in blockchain-enabled mobile edge computing systems," *IEEE Trans. Wireless Commun.*, vol. 19, no. 6, pp. 4321–4334, Jun. 2020.

[12] F. Guo, F. R. Yu, H. Zhang, H. Ji, M. Liu, and V. C. M. Leung, "Adaptive resource allocation in future wireless networks with blockchain and mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 1689–1703, Mar. 2020.

[13] 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, 2nd Quart. 2019.

[14] Z. Zhuang, J. Wang, Q. Qi, J. Liao, and Z. Han, "Adaptive and robust routing with lyapunov-based deep RL in MEC networks enabled by blockchains," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2208–2225, Feb. 2021.

[15] H. Chai, S. Leng, K. Zhang, and S. Mao, "Proof-of-reputation based-consortium blockchain for trust resource sharing in internet of vehicles," *IEEE Access*, vol. 7, pp. 175 744–175 757, 2019.

[16] 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 Trans. Veh. Technol.*, vol. 69, no. 4, pp. 4312–4324, Apr. 2020.

[17] X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access*, vol. 7, pp. 58 241–58 254, 2019.

[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 Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1189–1201, Apr.-Jun. 2021.

[19] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10 660–10 675, Dec. 2017.

[20] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11 049–11 061, Nov. 2018.

[21] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 2018.

[22] J. Zhao, M. Kong, Q. Li, and X. Sun, "Contract-based computing resource management via deep reinforcement learning in vehicular fog computing," *IEEE Access*, vol. 8, pp. 3319–3329, 2020.

[23] X. Chen, L. Zhang, Y. Pang, B. Lin, and Y. Fang, "Timeliness-aware incentive mechanism for vehicular crowdsourcing in smart cities," *IEEE Trans. Mobile Comput.*, to be published, doi:10.1109/TMC.2021.3052963.

[24] H. Liao, Y. Mu, Z. Zhou, M. Sun, Z. Wang, and C. Pan, "Blockchain and learning-based secure and intelligent task offloading for vehicular fog computing," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4051–4063, Jul. 2021.

[25] X. Lin, J. Wu, S. Mumtaz, S. Garg, J. Li, and M. Guizani, "Blockchain-based on-demand computing resource trading in IoV-assisted smart city," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 3, pp. 1373–1385, Jul.-Sep. 2021.

[26] K. Xiao, W. Shi, Z. Gao, C. Yao, and X. Qiu, "DAER: A resource preallocation algorithm of edge computing server by using blockchain in intelligent driving," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9291–9302, Oct. 2020.

[27] S. Iqbal, A. W. Malik, A. U. Rahman, and R. M. Noor, "Blockchain-based reputation management for task offloading in micro-level vehicular fog network," *IEEE Access*, vol. 8, pp. 52 968–52 980, 2020.

[28] C. Qiu, H. Yao, F. R. Yu, C. Jiang, and S. Guo, "A service-oriented permissioned blockchain for the internet of things," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 203–215, Mar./Apr. 2020.

[29] J. Luo, Q. Chen, F. R. Yu, and L. Tang, "Blockchain-enabled software-defined industrial internet of things with deep reinforcement learning," *IEEE Internet Things J.*, vol. 7, no. 6, pp. 5466–5480, Jun. 2020.

[30] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 13rd Symp. Oper. Syst. Des. Implementation*, 1999, pp. 173–186.

[31] C. Qiu, F. R. Yu, H. Yao, C. Jiang, F. Xu, and C. Zhao, "Blockchain-based software-defined industrial internet of things: A dueling deep Q -learning approach," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4627–4639, Jun. 2019.

[32] A. Clement, E. Wong, L. Alvisi, M. Dahlin, and M. Marchetti, "Making byzantine fault tolerant systems tolerate byzantine faults," in *Proc. 6th USENIX Symp. Networked Syst. Des. Implementation*, 2009, pp. 153–168.

[33] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Performance optimization for blockchain-enabled industrial internet of things (IIoT) systems: A deep reinforcement learning approach," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3559–3570, Jun. 2019.

[34] W. L. Tan, W. C. Lau, O. Yue, and T. H. Hui, "Analytical models and performance evaluation of drive-thru internet systems," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 1, pp. 207–222, Jan. 2011.

[35] J. Du, E. Gelenbe, C. Jiang, H. Zhang, and Y. Ren, "Contract design for traffic offloading and resource allocation in heterogeneous ultra-dense networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2457–2467, Nov. 2017.

[36] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[37] X. Huang, R. Yu, J. Kang, and Y. Zhang, "Distributed reputation management for secure and efficient vehicular edge computing and networks," *IEEE Access*, vol. 5, pp. 25 408–25 420, 2017.

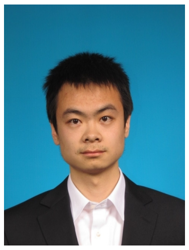
[38] M. Liu, F. R. Yu, Y. Teng, V. C. M. Leung, and M. Song, "Distributed resource allocation in blockchain-based video streaming systems with mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 695–708, Jan. 2019.

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, vol. 80, Jul. 2018, pp. 1861–1870.

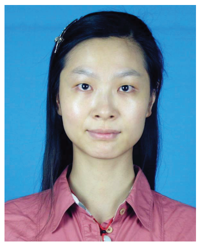
[40] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine, "Soft actor-critic algorithms and applications," *arXiv:1812.05905*, 2018.

[41] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. 13th AAAI Conf. Artif. Intell.*, Feb. 2016, pp. 2094–2100.

[42] J. Feng, F. Richard Yu, Q. Pei, X. Chu, J. Du, and L. Zhu, "Cooperative computation offloading and resource allocation for blockchain-enabled mobile-edge computing: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6214–6228, Jul. 2020.



Jinming Shi received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2015. He is currently working toward the Ph.D degree in electronic engineering with Tsinghua University, Beijing, China. His research interests include Internet of Vehicles, blockchain, vehicular fog computing, and deep reinforcement learning.



Jun Du received her B.S. in information and communication engineering from Beijing Institute of Technology, in 2009, and her M.S. and Ph.D. in information and communication engineering from Tsinghua University, Beijing, in 2014 and 2018, respectively. She currently holds a postdoctoral position with the Department of Electrical Engineering, Tsinghua University. Her research interests are mainly in resource allocation and system security of heterogeneous networks and space-based information networks.

Yuan Shen received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2005, and the S.M. degree in computer science and the Ph.D. degree in electrical engineering from MIT, Cambridge, MA, USA, in 2008 and 2014, respectively. He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include network localization and navigation, inference techniques, resource allocation, and cooperative networks.

Jian Wang (Senior Member, IEEE) received the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2006. In 2006, he joined the faculty of Tsinghua University, where he is currently a Professor with the Department of Electronic Engineering. His research interests include application of statistical theories, optimization, machine learning to communication, networking, navigation, and resource allocation problems.

Jian Yuan received his Ph.D. degree in electrical engineering from the University of Electronic Science and Technology of China, in 1998. He is currently a professor in the Department of Electronic Engineering at Tsinghua University, Beijing, China. His main research interest is in complex dynamics of networked systems.

Zhu Han (Fellow, IEEE) received the Ph.D. degree in electrical and computer engineering from the University of Maryland, College Park, in 2003.,He is a John and Rebecca Moores Professor with the Electrical and Computer Engineering Department as well as with the Computer Science Department, University of Houston, Houston, TX, USA. He is also a Chair Professor with National Chiao Tung University, Hsinchu, Taiwan.