A Blockchain-Based Distributed Machine Learning (BDML) Approach for Resource Allocation in Vehicular Ad-Hoc Networks

Dajun Zhang[1(✉)], Wei Shi[1], and Ruizhe Yang[2]

[1] Carleton University, 1125 Colonel By Drive, Ottawa, Canada
dajunzhang9038@gmail.com
[2] Beijing Laboratory of Advanced Information Networks, Beijing University of Technology, Beijing, China

Abstract. Recently, effective allocation of VANET resources is a key factor in promoting the development of VANETs. Due to high bandwidth costs, poor time efficiency, and a high risk of privacy leakage, the use of traditional centralized data centers to analyze massive data has proven to be a difficult task. These challenges have prompted a revolutionary change in VANET architectures to scatter computations from a centralized data center to distributed network edges. Distributed VANET configurations leverage the computing power of network edges by using a large number of mobile devices which frequently exchange data with the edge of the network or among themselves. However, the heterogeneity and distrust of the distributed edge hinder the efficient, reliable, and secure allocation of VANET resources. In this paper, we express the allocation strategy for both computing and network resources as a joint optimization problem. We use a local deep reinforcement learning with a prioritized experience replay mechanism on edge nodes and use the blockchain for sharing the optimal learning results to optimize the overall resource allocation problem. Simulation results show that our proposed scheme is superior to a current machine learning approach.

Keywords: Vehicular ad hoc networks · Blockchain · Deep reinforcement learning

1 Introduction

Recently, vehicular ad-hoc networks (VANETs) have aroused great interest. In the VANET environment, the limitations of network resources prevent the improvement of edge computing. Currently, due to privacy concerns and the frequent data loss during high-speed streaming data transmission, data shared between VANET users are limited to only emergency situations. Furthermore, the limitation of edge computing power brings new challenges and difficulties to resource allocation in VANETs [1]. For example, the uncertainty of vehicle movement may lead to competition between VANET nodes (i.e. the computing

C. Yu et al. (Eds.): GPC 2022, LNCS 13744, pp. 110–121, 2023.
https://doi.org/10.1007/978-3-031-26118-3_8

unit on board of a vehicle) for the computing power of edge nodes (i.e. RSUs) resulting in the insufficient allocation of computing capability. Considering the sparse deployment of RSUs, when the number of vehicles within the coverage of a RSU increases rapidly, the resource spectrum of the RSU cannot meet the needs of all vehicles. Therefore, fair and effective allocation of computing resources and network resources to VANET nodes has become an urgent problem to be solved.

At this stage, some scholars have put forward resource allocation mechanisms in VANETs. The research [2] proposed the allocation of computing resources for video processing, aiming at improving the quality of service for users. The authors of [3] proposed a VANET framework for the dynamic adjustment of network, cache, and computing resources, aiming to improve the joint resource allocation problem of existing VANETs. The authors of [4] designed a spectrum-sharing scheme in order to solve the data conflict between the cellular network of vehicles and users. For the occupation of unlicensed channels, the two can fairly compete for the right to use the channel. In [5], the author proposes a new decomposition algorithm that utilizes integer linear programming to fully allocate computing resources. It provides ideas for the future software-defined VANET data-sharing architecture. In order to ensure the correct deployment of the blockchain in the distributed software-defined Internet of Vehicles, the authors of [6] propose a distributed software-defined VANET architecture based on the blockchain to ensure the utility of resource allocation.

In this article, we propose an integrated resource allocation scheme for the VANETs based on the blockchain and Deep Reinforcement Learning (DRL). The contributions of this article are summarized as follows:

- We propose a new resource (network resource and computing capability) allocation scheme based on blockchain. Each VANET node trains a neural network locally through the DRL algorithm to obtain resource allocation strategies and shares the best local learning results through distributed edge nodes. Here, we aim to use the communication utility to characterize the quality of the communication link between the node and the RSU, so as to allocate the best RSU to neighboring nodes for local training. Meanwhile, the computing utility is used to characterize the utilization of computing resources. The VANET node trains the joint resource utility locally within a fixed time, and then uploads the result to the blockchain system, aiming to improve the joint resource utility through the parameter sharing of the blockchain system. The core advantage of using the blockchain for sharing is that the blockchain is a completely decentralized system, and the local training results are stored in the transaction. The non-tamperable feature of the blockchain ensures the safety of local training results. The proposed scheme lays a new foundation for intelligence sharing between edges.
- We make improvements to the traditional DRL algorithm in local training. Aiming at solving the problem that the traditional DRLs experience in inefficient random sampling using replay strategy, we have introduced a prioritized experience replay strategy to the local training. We define the priority of the

samples and prioritize the samples in the experience replay pool to improve the efficiency of local training.
– Finally, we have adopted a Redundant Byzantine Fault-Tolerant (RBFT) [7] mechanism. Since a permissioned-blockchain system is used, RBFT has better fault tolerance and better overall performance than the Practical Byzantine Fault-Tolerant (PBFT) consensus mechanism.

The rest of the paper is organized as follows: the system model is described in Sect. 2. Section 3 illustrate the problem formulation of the local training. In Sect. 4, we introduce the local training process and blockchain sharing mechanism. In Sect. 5, we present the experiments that have been conducted in this work. Finally, conclusions are described in Sect. 6.

2 System Description

In this section, we illustrate the system model in our proposed framework. We discretize the time $\mathcal{T}$ into a number of time slot $\mathcal{T} = \{1, 2, ..., t, ..., T\}$.

2.1 Communication Model

Figure 1 shows that we use collaborative edge computing to balance the loss of high-speed streaming data transmission and the overhead on transmission. The proposed framework has $\mathcal{C} = \{R_1, R_2, ..., R_c, ..., R_C\}$ Road Side Units (RSUs) and $\mathcal{E} = \{M_1, M_2, ..., M_e, ..., M_E\}$ mobile edge servers. In our proposed architecture, U vehicles around multiple RSUs and edge servers communicate with each other, denoted by $\mathcal{U} = \{V_1, V_2, ..., V_u, ..., V_U\}$. In this paper, we implement blockchain sharing of local training results (communication and computation utilization) of multiple vehicles.

The communication model is defined as finite-state Markov channels (FSMCs) [8]. The channel state $\delta^{V_u,R_c}(t)$ between V_u and R_c depends on the received signal-to-noise ratio (SNR), which is divided into K levels that is defined as $\mathcal{F} = \{F_0, F_1, ..., F_{k-1}, ..., F_K\}$.

Since the communication channel is a time-varying channel, we can set the variation of SNR follow a Markov decision process. Let $p_{k,k'} = Pr\{\delta^{V_u,R_c}(t+1) = F_{k'}|\delta^{V_u,R_c}(t)\}$ be the channel state transition probability, where $k, k' \in \mathcal{K}$. Hence, $\mathcal{P} = [p_{k,k'}]_{K\times K}$ is the channel state transition matrix of the $K \times K$ dimension.

We define the available bandwidth allocated to V_u as $H_{R_c}^{V_u}$. According to Shannon's theorem, the maximum communication rate $r_{R_c}^{V_u}(t)$ is:

$$r_{R_c}^{V_u}(t) = H_{R_c}^{V_u}(t)\log_2(1 + \delta^{V_u,R_c}(t)) \tag{1}$$

Therefore, we define the communication resource utilization of V_u as the ratio of $r_{R_c}^{V_u}(t)$ to the available bandwidth $H_{R_c}^{V_u}$ allocated to V_c. Hence, we have:

$$D_{V_u}^{Comm}(t) = \sum_{R_c=1}^{R_C} a_{V_u}^{R_c}(t)\frac{\sigma_{V_u} r_{R_c}^{V_u}(t)}{\theta_{R_c} H_{R_c}^{V_u}(t)} \tag{2}$$

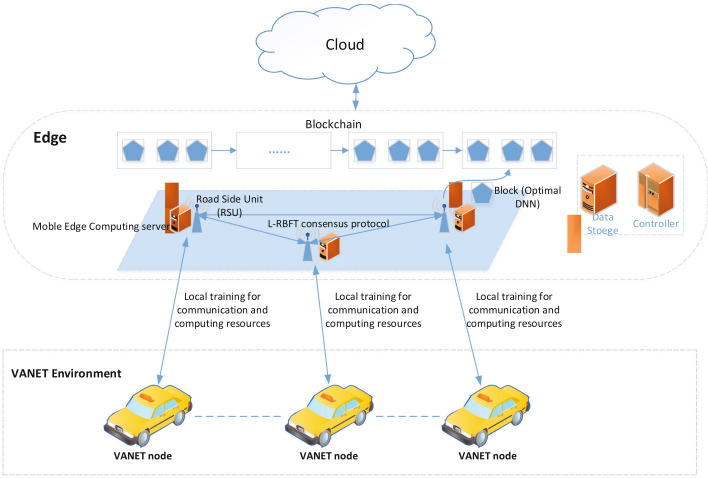

Fig. 1. Blockchain-based resource allocation framework of VANETs.

where σ_{V_u} means the revenue per unit communication rate that V_u can achieve, and θ_{R_c} denotes the unit payment for using networking resources from R_c. We will illustrate the action value $a_{V_u}^{R_c}(t)$ in Sect. 3.

2.2 Computation Model

We use $\epsilon_{V_u}^{M_e}(t)$ to represent the computing capability of each M_e assign to V_u at time slot t, which satisfy the Markov decision process. We discretize random variable $\epsilon_{V_u}^{M_e}(t)$ into Z state, indexed by $\Lambda = \{\Lambda_0, \Lambda_1, ..., \Lambda_{Z-1}\}$. The state transition probability $g_{z'z} = Pr\{\epsilon_{V_u}^{M_e}(t+1) = \Lambda_z | \epsilon_{V_u}^{M_e}(t) = \Lambda_{z'}\}$ determines the state change of $\epsilon_{V_u}^{M_e}(t)$, where $\Lambda_z, \Lambda_{z'} \in \Lambda$.

We usually use time delay to characterize the Quality of Service (QoS) in VANETs. We quantify QoS in terms of queuing delay and propagation delay [9]. We refer the queuing model proposed in [10], we have:

$$T_{qd}^{V_u} = \frac{1}{\epsilon_{V_u}^{M_e}(t) - \lambda_{V_u}(t)} - \frac{1}{\lambda_{V_u}(t)} \cdot \frac{Q_L^{V_u} \rho^{Q_L^{V_u}}}{1 - \rho^{Q_L^{V_u}}} \tag{3}$$

where workload arrival rate is denoted as $\lambda_{V_u}(t)$, $\rho^{Q_L^{V_u}} = \lambda_{V_u}(t)/\epsilon_{V_u}^{M_e}(t)$ is the utilization, and the maximum queue length is $Q_L^{V_u}$.

The propagation delay T_{pd} can be defined as:

$$T_{pd}^{V_u} = d_{V_u}^{M_e}/\gamma \tag{4}$$

where γ is propagation speed and $d_{V_u}^{M_e}$ is the physical distance between $V_u(x_u, y_u)$ and $M_e(x_e, y_e)$. Hence, $d_{V_u}^{M_e}$ can be defined as $d_{V_u}^{M_e} = \sqrt{(x_e - x_u)^2 + (y_e - y_u)^2}$.

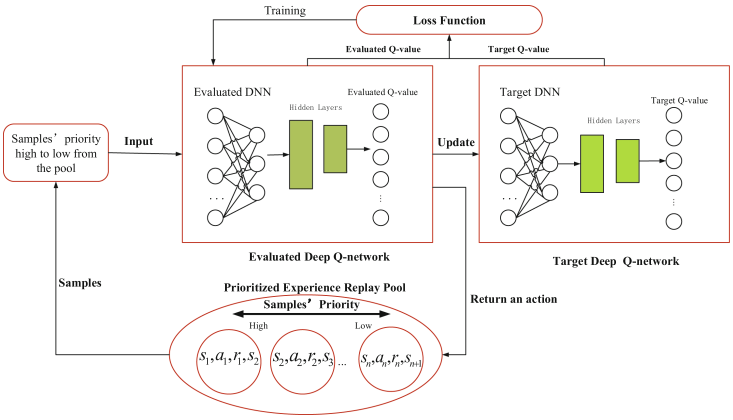

Fig. 2. The workflows of local training.

Finally, we use $T_{V_u} = T_{qd}^{V_u} + T_{pd}^{V_u}$ to characterize the service delay of the overall system.

Therefore, the computing resource utilization $D_{V_u}^{Comp}(t)$ is:

$$D_{V_u}^{Comp}(t) = \sum_{M_e=1}^{M_E} a_{V_u}^{M_e}(t) \frac{\zeta_{V_u} \lambda_{V_u}}{\alpha_{V_u} k_{M_e} o_{V_u} + \beta_{V_u} T_{V_u}} \tag{5}$$

where α_{V_u} and β_{V_u} are the weight ratios of relevant parameters. Here, ζ_{V_u} represents the benefits that V_u can obtain due to the workload of operating unit, k_{M_e} is the operating overhead per unit CPU cycle, and o_{V_u} is the number of CPU cycles required for the computation task. We will illustrate the meaning of action value $a_{V_u}^{M_e}(t)$ in Sect. 3.

3 Problem Formulation

In this section, we discuss the resource allocation problem of VANET node in detail by defining the state space, action space and reward function.

3.1 System State

We set a parameter $s(t), t \in \mathcal{T}$ represent the system state space of proposed framework at time slot t. $s(t)$ includes two components: $\delta^{V_u,R_c}(t)$ and $\epsilon_{V_u}^{M_e}(t)$. Here, we have:

$$s(t) = \begin{bmatrix} \delta^{V_1,R_1}(t) & \cdots & \delta^{V_1,R_c}(t) & \cdots & \delta^{V_1,R_C}(t) \\ \delta^{V_2,R_1}(t) & \cdots & \delta^{V_2,R_c}(t) & \cdots & \delta^{V_2,R_C}(t) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \delta^{V_U,R_1}(t) & \cdots & \delta^{V_U,R_c}(t) & \cdots & \delta^{V_U,R_C}(t) \\ \epsilon_{V_1}^{M_1}(t) & \cdots & \epsilon_{V_1}^{M_e}(t) & \cdots & \epsilon_{V_1}^{M_E}(t) \\ \epsilon_{V_2}^{M_1}(t) & \cdots & \epsilon_{V_2}^{M_e}(t) & \cdots & \epsilon_{V_2}^{M_E}(t) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \epsilon_{V_U}^{M_1}(t) & \cdots & \epsilon_{V_U}^{M_e}(t) & \cdots & \epsilon_{V_U}^{M_E}(t) \end{bmatrix} \tag{6}$$

3.2 System Action

We set $a(t), t \in \mathcal{T}$ as the action space of the system. The definition of $a(t)$ is:

$$a(t) = \begin{bmatrix} a_{V_1}^{R_1}(t) \cdots a_{V_1}^{R_c}(t) \cdots a_{V_1}^{R_C}(t) \\ a_{V_2}^{R_1}(t) \cdots a_{V_2}^{R_c}(t) \cdots a_{V_2}^{R_C}(t) \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\ a_{V_U}^{R_1}(t) \cdots a_{V_U}^{R_c}(t) \cdots a_{V_U}^{R_C}(t) \\ a_{V_1}^{M_1}(t) \cdots a_{V_1}^{M_e}(t) \cdots a_{V_1}^{M_E}(t) \\ a_{V_2}^{M_1}(t) \cdots a_{V_2}^{M_e}(t) \cdots a_{V_2}^{M_E}(t) \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\ a_{V_U}^{M_1}(t) \cdots a_{V_U}^{M_e}(t) \cdots a_{V_U}^{M_E}(t) \end{bmatrix} \tag{7}$$

where $a_{V_u}^{R_c}(t)$, and $a_{V_u}^{M_e}(t)$ are:

1) The value of $a_{V_u}^{R_c}(t)$ is 1 or 0. For example, if the action value of R_c is 1 at time t, then RSU and V_c communicate with each other. Otherwise, the action value of R_c is 0. In this paper, we assume that V_u can only communicate with one RSU in a time slot. So we can get $\sum_{R_c=1}^{R_C} a_{V_u}^{R_c}(t) = 1$.
2) The value of $a_{V_u}^{M_e}(t)$ is $\{0, 1\}$. For example, if $a_{V_u}^{M_e}(t) = 0$ at time slot t, then M_e does not need to calculate the computing task from V_u. Otherwise, the action value of $a_{V_u}^{M_e}(t)$ is 0. We assume that only one M_e handles the computation task in a time slot. So we can get $\sum_{M_e=1}^{M_E} a_{V_u}^{M_e}(t) = 1$.

3.3 Reward Function

The reward function is accomplished through decision-making in $a(t)$ and $s(t)$. Then, the immediate reward function (utility) of the system is:

$$r_{V_u}(t) = \sum_{V_u=1}^{V_U} (D_{V_u}^{Comm}(t) + D_{V_u}^{Comp}(t)) \tag{8}$$

4 Blockchain-Based Sharing Strategy

In this section, we describe the local training strategy of VANET nodes and the details of the blockchain-based sharing mechanism.

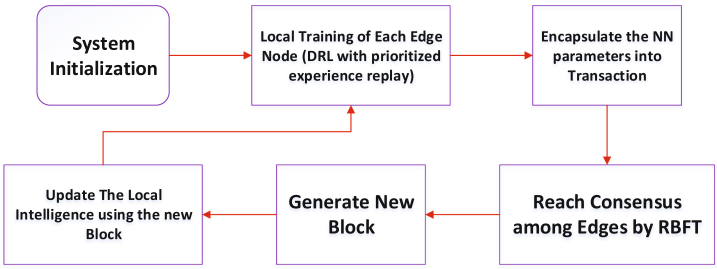

Fig. 3. The sharing mechanism used in BDML

4.1 Local DRL Training with Prioritized Experience Replay of VANET Node

In the deep reinforcement learning experience replay pool, random sampling may repeatedly select redundant samples and reduce training efficiency. Therefore, we have added the Prioritized Experience Replay mechanism (PER) to local learning. The core idea of PER is to prioritize samples. In other words, this method can find samples more efficiently. Figure 2 shows the local training process of the proposed framework. In PER, the probability of each sample (e.g., $(s(t),\ a(t),\ r_{V_u}(t),\ s(t+1))$ being selected is monotonic according to priority. Specifically, the probability of extracting the transition marked j is defined as:

$$P(j) = \frac{p_j^\tau}{\sum_k p_k^\tau} \tag{9}$$

where p_j^τ is the priority of sample j.

In PER, temporal-difference (TD) errors are often used to prioritized samples [11]. In other words, the premise that a sample has higher priority for the agent to learn is that the TD error is larger. Therefore, we have:

$$\rho_j = R_j + \gamma_j Q_{target}\left(S_j, \arg\max Q(S_j, a)\right) - Q(S_{j-1}, A_{j-1}) \tag{10}$$

where R_j and $Q(S_j, a)$ are the reward value and Q-value of sample j respectively.

After we determine the TD error j, we need to define importance-sampling weight (IS-weight) as:

$$w_j = \left(\frac{1}{G} \cdot \frac{1}{P(j)}\right)^\kappa \tag{11}$$

where κ is responsible bias adjustment and G represents the size of experience replay memory.

4.2 Blockchain Sharing Mechanism

In our proposed architecture, RSUs and internal edge computing nodes form the basic structure of the blockchain. The block contains a batch of transactions, and

the consensus mechanism is responsible for unified packaging and sequencing. After receiving a block, the blockchain node executes transactions in sequence based on the original account status and reads/writes the status data of the relevant account during this period. The completion of a transaction execution means that the state of the blockchain has undergone a change.

For each transaction, there will be a corresponding transaction receipt or illegal transaction record in the blockchain to indicate the final execution result. If the transaction is a legal transaction, after the execution ends, the result of the transaction execution will be recorded in the transaction receipt. Otherwise, the reason for the error will be recorded in an illegal transaction record. For every illegal transaction, the error information will be encapsulated into an illegal transaction record and stored locally on the node. In addition to the transaction data related to it, the illegal record will also record the specific error reason. For example, if a malicious VANET node sends a false local training result, it will not have sufficient authority to execute the smart contract.

The transaction is initiated by an external user (VANET node). The locally trained VANET node encapsulates the training results (DNN parameters and the training loss) in the transaction and uploads them to the nearby RSU node. When the blockchain node receives the transaction and verifies it first, the node will only process the verified request. The node will do the following transaction verification: (1) Verify the legality of the transaction field, including the transaction format and the legality of the timestamp; (2) Whether the same transaction has been submitted; (3) Verify the transaction signature. After the transaction passes the above verification, a consensus is reached between the blockchain nodes, and the received transaction is broadcast to the consensus nodes of the entire network.

1) *Pre-Prepare*: The Pre-Prepare consensus master node will sequence the transactions within a certain period of time (or a certain number), package them into a block, and then send them to the entire network for consensus.
2) *Preparation*: Prepare all consensus nodes preprocess the block and broadcast the resulting hash.
3) *Submit*: Commit all consensus nodes to write to the block and update the blockchain ledger.

Illegal transactions discovered during the execution process will be stored in the illegal transaction records of the database, and will not be recorded on the blockchain ledger. All legal transactions are stored on the blockchain ledger. When consensus is reached, the node verifies that the transaction and block are correct and valid. The node will automatically execute the smart contract. The node uses its own private key to sign the content of the transaction initiator and the transaction receiver to prevent the content of the transaction from being tampered with. After verifying the signature, MAC, and smart contract, the system sends the newly generated block back to the edge learning node and attaches the block to the blockchain. The edge learning node analyzes the payload information, that is, each edge learning node learns the parameters of

Table 1. Simulation parameters

Simulation parameter	Assigned value
Number of vehicles	8
σ_{V_u}	8 units/Mbps
θ_{R_c}	2 units/Mbps
$H_{R_c}^{V_u}$	4 Mbps
λ_{V_u}	90 Mbps
ζ_{V_u}	3 unit/Mbps
α_{V_u}	0.5
β_{V_u}	0.5
k_{M_e}	2 w/Mcycles
o_{V_u}	50 Mcycles

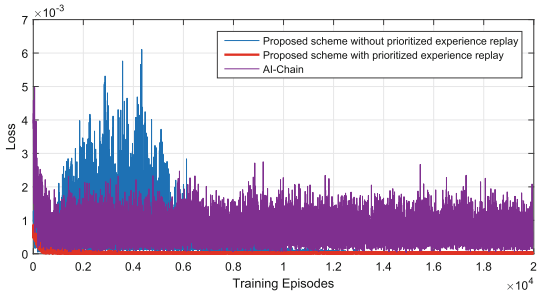

Fig. 4. The convergence performance comparison of three schemes.

the deep neural network from other learning nodes to share the optimal neural network parameters. Figure 3 shows the procedure of the blockchain sharing mechanism.

5 Simulation Results and Discussions

In this section, we verify the superiority of the proposed framework through Tensorflow [12] simulation. Deep Q-learning with a prioritized experience replay mechanism is adopted in the local training process.

5.1 Simulation Setup

Here, we need to point out in particular that the communication utility and computing utility are the key elements that determine the system resource allocation. The communication utility model determines the RSU assigned to a certain VANET node and the channel state of this node. The computing utility model determines the computing capability of the edge computing server

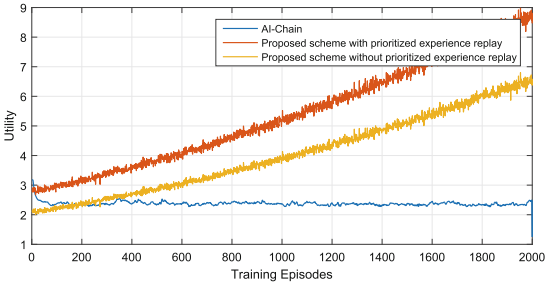

Fig. 5. The utility comparison of three schemes.

allocated to the VANET node for local training. Specifically, RSUs with better channel conditions and higher-power edge computing servers should be used to perform computing tasks. We use these two factors as the local training reward function to characterize the local training effect. Share the parameters of the local training model through the blockchain to enhance the utility of the overall system. Therefore, we treat the problem of joint resource allocation as proof of the effectiveness of our proposed architecture.

In our simulation, we use AI-chain [13] as a comparison scheme to demonstrate the superiority of our proposed scheme. Meanwhile, we have made some adjustments to the existing AI-chain. First, we apply AI-chain to the VANET environment. Secondly, we replaced the neural network for the local training part of AI-chain from the traditional deep neural network to Convolutional Neural Network (CNN). The purpose of this is to meet the comparison requirements of our simulation. The configuration of the AI-chain blockchain adopts a consensus mechanism called learning proof (permissionless), and the local training process follows the traditional deep neural network. The basic idea of AI-chain is to share the optimal neural network parameters among edge nodes through the consensus mechanism of learning proof to meet the needs of joint resource allocation. Therefore, [13] meets the requirements as a comparison scheme. Table 1 shows the simulation parameters in this paper.

5.2 Simulation Results

Figure 4 shows the fitting effect of the convergence curve characterized by the loss function under different schemes. We can see that the curve oscillates significantly at the beginning of the gradient descent. After a training period, the amplitude of the curve oscillations tends to flatten, which means that the optimal policy will be found when the training curve converges. The blockchain shared resource allocation scheme we proposed is different from DRL in that we use multiple edge computing nodes as providers of training results, and solve the joint resource allocation problem of the network by sharing learning results. Through these comparisons, the convergence performance of the blockchain shared resource allocation scheme we proposed has a better conver-

gence effect, which means that our scheme can obtain the optimal strategy faster after a short training, as shown in the red curve in Fig. 4 shown. The reason for the rapid convergence is that the edge computing nodes obtain better DNN parameters from other consensus nodes through the shared parameter mechanism of the blockchain. The advantage of the sharing mechanism is to reduce the waste of computing resources during the training process. In addition, the rapid convergence of the red curve shows that with the help of the sharing mechanism, the local training node iterates the actual Q-value faster, and it is easier to obtain the optimal joint resource allocation strategy, saving a lot of training time. In addition, the prioritized experience replay mechanism is better than the scheme without it. This is because the priority experience replay strategy sets the priority of the system samples, and the system trains according to the priority of the samples. It avoids the drawbacks of repeated training of samples in the experience replay pool due to random sampling, thereby reducing the number of operations and further optimizing the convergence performance.

Figure 5 depicts the fitting of the training curve tracking the reward function under different schemes. Specifically, Eq. (8) captures the calculation method of system utility, which is used to measure the performance of the two resource allocation schemes used in this work. As shown in the figure, the red curve gets the highest reward, which means that our proposed combination of blockchain sharing and priority experience replay will have the best utility while converging quickly. Moreover, with the increase of events, the utility of the red curve has been significantly improved. This is because the system converges quickly and improves the utility of computing resources. As the number of elements in the state space increases, the environment becomes more complex, resulting in a decrease in convergence performance, so the ascent of the curve slows down as the plot increases. However, in the case of any number of features, the joint consideration of network resource and computing resource allocation can allocate more powerful edge computing servers for users, so the resource utilization rate has been significantly improved compared with traditional DRL resource allocation schemes.

6 Conclusions and Future Work

In this article, we proposed a novel joint resource allocation scheme based on sharing mechanism with blockchain for future Internet of Vehicles. We first executed the local deep reinforcement learning algorithm with a prioritized experience replay mechanism on each edge node. Then, each edge node shared its learning result with others via blockchain in order to optimize the joint resource allocation problem. Most importantly, we considered channel resources and computing resources as components of a joint resources allocation strategy. Considering the importance of the samples in the experience replay pool, we used a prioritized experience replay to accelerate the local training speed. This approach can significantly improve the training efficiency through the simulation results. Incorporation of intrusion detection systems will be considered in the future.

Acknowledgements. We gratefully acknowledge the financial support from the Natural Sciences and Engineering Research Council of Canada (NSERC) under Grants No. RGPIN-2020-06482.

References

1. Liu, J., Wan, J., Zeng, B., Wang, Q., Song, H., Qiu, M.: A scalable and quick-response software defined vehicular network assisted by mobile edge computing. IEEE Commun. Mag. **55**(7), 94–100 (2017)
2. Zhang, Z., Wang, R., Yu, F.R., Fu, F., Yan, Q.: QoS aware transcoding for live streaming in edge-clouds aided HetNets: an enhanced actor-critic approach. IEEE Trans. Veh. Technol. **68**(11), 11295–11308 (2019)
3. He, Y., Yu, F.R., Zhao, N., Yin, H., Boukerche, A.: Deep reinforcement learning (DRL)-based resource management in software-defined and virtualized vehicular ad hoc networks. In: Proceedings of the 6th ACM Symposium on Development and Analysis of Intelligent Vehicular Networks and Applications (DIVANet 2017), New York, NY, USA, 47C54 (2017)
4. Wang, P., Di, B., Zhang, H., Bian, K., Song, L.: Cellular V2X communications in unlicensed spectrum: harmonious coexistence with VANET in 5G systems. IEEE Trans. Wireless Commun. **17**(8), 5212–5224 (2018)
5. Luo, G., et al.: Software-defined cooperative data sharing in edge computing assisted 5G-VANET. IEEE Trans. Mob. Comput. **20**(3), 1212–1229 (2021)
6. Zhang, D., Yu, F.R., Yang, R.: Blockchain-based distributed software-defined vehicular networks: a dueling deep Q -learning approach. IEEE Trans. Cognit. Commun. Networking **5**(4), 1086–1100 (2019)
7. Aublin, P.-L., Mokhtar S.B., Quçma, V.: RBFT: redundant byzantine fault tolerance. In: 2013 IEEE 33rd International Conference on Distributed Computing Systems, pp. 297–306 (2013)
8. He, Y., Zhao, N., Yin, H.: Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach. IEEE Trans. Veh. Technol. **67**(1), 44–55 (2018)
9. Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, F.R., Han, Z.: Computing resource allocation in three-tier IoT fog networks: a joint optimization approach combining stackelberg game and matching. IEEE Internet Things J. **4**(5), 1204–1215 (2017)
10. Tian, J., Han, Q., Lin, S.: Improved delay performance in VANET by the priority assignment. In: IOP Conference Series: Earth and Environmental Science, vol. 234, no. 1 (2019)
11. Schaul, T., Quan, J., Antonoglou, I., et al.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)
12. Abadi, M., Agarwal, A., Barham, P., et al.: Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467 (2016)
13. Qiu, C., Yao, H., Wang, X., Zhang, N., Yu, F.R., Niyato, D.: AI-chain: blockchain energized edge intelligence for beyond 5G networks. IEEE Network **34**(6), 62–69 (2020)