Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 3207456, 8 pages
https://doi.org/10.1155/2022/3207456

Research Article

An Efficient Resource Management Optimization Scheme for Internet of Vehicles in Edge Computing Environment

Anqing Zhu and Youyun Wen

Management School, South China Business College Guangdong University of Foreign Studies, Guangzhou, Guangdong 510000, China

Correspondence should be addressed to Anqing Zhu; zhuaq18@126.com

Received 2 April 2022; Revised 26 April 2022; Accepted 11 May 2022; Published 28 May 2022

Academic Editor: Le Sun

The contradiction between limited network resources and a large number of user demands in vehicle environment will cause a lot of system delay and energy consumption. To solve the problem, this paper proposes an efficient resource management optimization scheme for Internet of Vehicles in edge computing environment. Firstly, we give a detailed formulation description of communication and computing cost incurred in the resource optimization process. Then, the optimization objective of this paper is clarified by considering the constraints of computing resources, and system delay and energy consumption are considered comprehensively. Secondly, considering dynamic, random, and time-varying characteristics of vehicle network, the optimal resource management scheme of Internet of Vehicles is given by using distributed reinforcement learning algorithm to optimize total system overhead to the greatest extent. Finally, experiments show that when bandwidth = 40 MHz, the total system cost of the proposed algorithm is only 3.502, while that of comparison algorithms is 4.732 and 4.251, respectively. It is proved that the proposed method can effectively reduce the total system overhead.

1. Introduction

In recent years, the automotive industry has developed rapidly, and intelligence and networking have become an important trend in the future development of automotive industry [1]. On the one hand, these technologies enable communication and information exchange between Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I), helping to build safe, collaborative, and intelligent transportation systems. On the other hand, this in turn generates a large amount of data, and at the same time, the demand for computing, communication, and content increases significantly [2–4]. With the development of Internet of Vehicles (IoV) and intelligent connected vehicles, in-vehicle infotainment applications such as road safety, intelligent navigation, autonomous driving, and in-vehicle entertainment continue to emerge. This promotes the development of intelligent transportation and brings a great improvement to driving experience [5–7]. Due to the particularity of the physical location of vehicles and cloud servers, the backhaul link capacity is limited. Such a high content demand of the Internet of Things will bring a huge burden to the core network [9]. At the same time, they also pose a major challenge to support massive content delivery and meet the low-latency requirements of IoT [10–12].

The introduction of mobile edge computing (MEC) technology makes up for the network instability and delay limitations of cloud computing in IoT scenario and is more suitable for low-latency, high-reliability task computing on IoT requirements [13–18]. The cloud server located in core network is far away from vehicles, and vehicles need to rely on a large base station for multi-hop transmission to offload tasks to the cloud server for processing. However, it is prone to network fluctuations and transmission interruptions and is unreliable for in-vehicle applications, especially safe driving applications [19, 20]. Therefore, using distributed MEC services to replace traditional cloud computing services can effectively solve the resource management optimization problem in IoT [21].

The main factors that affect the decision of computing offloading are the execution delay and energy consumption of task. Thus, optimization goals usually include solutions such as reducing delay, reducing energy consumption, and weighting between delay and energy. Reinforcement learning can capture the hidden dynamics of environment well, so it is often used to optimize resource allocation algorithms. Liu et al. [22] proposed a resource allocation strategy based on deep reinforcement learning (DRL). Zhan et al. [23] designed a strategy optimization method based on DRL by using game theory. Huang et al. [24] studied the wireless charging MEC network and proposed an online decision-making method based on DRL. Hui et al. [25] proposed a content dissemination framework based on edge computing. Combining the selfishness and transmission ability of vehicles, the authors designed a two-level relay selection algorithm to reasonably select relay vehicles to meet different transmission needs. Su et al. [26] used the vehicles parked around the street and the vehicles driving along the road to communicate the vehicle social community through V2V and used the content cached in the parked vehicles to reduce the delay of content download. Zhao et al. [27] proposed a caching strategy in V2V scenario with information as the center and designed a dynamic probabilistic caching scheme. Zhang et al. [28] proposed a MEC scenario computing resource allocation scheme based on DRL network, which avoids falling into the disaster of dimensionality. Zhang et al. [29] proposed a joint optimization scheme of IoT content caching and resource allocation based on MEC in high-speed free-flow scenario, which reduced data acquisition latency. Li [29] proposed a resource allocation strategy for computing unloading in vehicle Internet based on DRL. However, in the case of limited network resources and a large number of user demands, the above research has the problems of excessive resource delay consumption and large energy consumption. Therefore, the optimization of IoT resource management in the MEC system scenario is a challenging problem.

Based on the above analysis, in view of delay and energy consumption caused by the contradiction between limited network resources and a large number of user needs in vehicle environment, this paper proposes an efficient resource management optimization scheme for IoT in edge computing environment. This method takes minimizing the weighted sum of system delay and energy consumption as the optimization goal and constructs a communication model and task offloading optimization model in IoT edge computing scenario. Moreover, a solution algorithm based on distributed reinforcement learning is used to optimize the total system overhead.

2. System Model and Problem Modeling

2.1. System Model. The system model is shown in Figure 1. J road side units (RSUs) are evenly distributed on the road, and all have MEC services configured, denoting MEC server as mec_j, $j \in \{1, 2, \ldots, J\}$. Each of C randomly distributed vehicles performs multiple computing tasks. Suppose the sum of computing tasks of all vehicles is N, and the computing

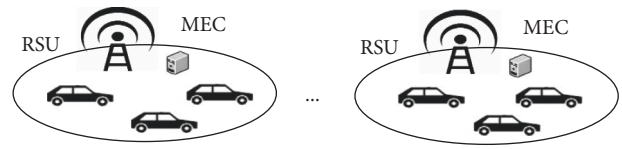

Figure 1: IoT system model.

tasks are denoted by L. b represents the size of input data, w represents the task computation amount, and $t^{\max}$ represents the task deadline. If the task processing exceeds the time limit, it means that the task processing fails, and R_L represents MEC cell carried by vehicle-mounted terminal to which the task belongs. ω represents the importance of computing task to distinguish the task is a secure computing task and a common computing task. Therefore, the computational task can be denoted as $L = \{b, w, \omega, t^{\max}, R_L\}$. Let i represent the number of onboard terminals offloading the computing task to MEC server, and $x = 0$ indicates that the task is executed locally. The offloading strategies of N computing tasks constitute the offloading strategy vector set $X = \{x_1, x_2, \ldots, x_N\}$.

2.2. Communication Model. When task i chooses to perform computing offloading, a corresponding offloading decision needs to be made to decide which MEC server to offload to and which channel to select to upload data. When the offloading decision vector $\mathbf{d}$ of all users is given, data transmission rate $R_n^m(\mathbf{d})$ on the n channel between user u_i of offloading decision $d_n^0 > 0$ and the j RSU can be obtained. The maximum information transfer rate V is

$$V = W \log_2(1 + \text{SNR}), \tag{1}$$

where W is the channel bandwidth and SNR is the ratio of average power of signal transmitted in the channel to noise power in the channel, that is, the signal-to-noise ratio. Its calculation is

$$\text{SNR} = \frac{p_i g_i^h}{\gamma^2 + \sum_{i \in V} p_i g_i^h}, \tag{2}$$

where p_i represents the transmission power of users, that is, the transmit power of user equipment; g_i^h represents the channel gain of communication channel selected by users; and γ^2 represents the white Gaussian noise power.

The data transfer rate $V_{i,j}^h(d)$ is

$$V_{i,j}^h(d) = B \log_2\left(1 + \frac{p_i g_i^h}{\gamma^2 + \sum_{i \in V} p_i g_i^h}\right). \tag{3}$$

2.3. Computing Offloading Model

2.3.1. Local Computing. Assuming that task i is only calculated locally, only the calculation delay is considered. e^{loc} represents the computing capability of vehicle terminals. The processing delay of local tasks and the energy required for local computing are expressed as

$$t_i = t_{\text{loc}} = \frac{w_i}{e^{\text{loc}}},$$
$$E_i = t_i P_{\text{loc}}. \tag{4}$$

When MEC server resources are insufficient, the system unloads the task to other servers.

2.3.2. Local Server Computing. When the vehicle communicates directly with the local server, the vehicle will unload the computing task to the server in the cell. After the server completes the execution, the result will be returned to the vehicle immediately. The total task delay includes upload delay, server calculation delay, and return delay. Let $t_{i,j}^{\text{mec}}$ denote the task execution delay, $e_{i,j}^{\text{mec}}$ denote the computing resources, and $v_{i,j}$ denote the wireless transmission rate:

$$t_{i,j}^{\text{mec}} = \frac{w_i}{e_{ij}^{\text{mec}}},$$
$$t_{i,j}^{\text{trans}} = \frac{b_i}{v_{ij}}. \tag{5}$$

Since the return rate is much higher than the upload rate, the return delay of the calculation result can be ignored. The total time delay of unloading calculation is

$$t_{i,j} = t_{i,j}^{\text{trans}} + t_{i,j}^{\text{mec}}. \tag{6}$$

The energy consumption for unloading calculation is

$$E_{i,j} = P_{\text{loc}}\left(t_{i,j}^{\text{trans}} + t_{i,j}^{\text{mec}}\right), \tag{7}$$

where P_{loc} represents the energy consumption per unit cycle of local CPU.

2.3.3. Other Server Computing. When the MEC server in the cell where the vehicle is located is overloaded, the computing task is unloaded to other cell servers. Communication between MEC servers is generally performed through wired communication links such as optical fibers. Assuming that the average task transmission delay on the wired link is t_w and c represents the number of wired link hops between computing tasks offloaded to other servers, the task processing delay at this time is expressed as

$$t_{i,j}^{o} = 2ct_w + t_{i,j}^{\text{trans}} + \frac{w_i}{e_{i,j}^{\text{mec}}}. \tag{8}$$

Then, the energy consumption of other servers' offloading computing is

$$E_{i,j}^{o} = P_{\text{loc}}\left(2ct_w + t_{i,j}^{\text{trans}} + \frac{w_i}{e_{i,j}^{\text{mec}}}\right). \tag{9}$$

2.4. Problem Modeling. τ is the calculated weight, and the weighted sum of the total delay is

$$t_{\text{all}} = \sum_{i=1}^{N}\left(\tau t_i + (1-\tau)\sum_{j=1}^{J} t_{ij}\right). \tag{10}$$

The total energy consumption is

$$c_{\text{all}} = \sum_{i=1}^{N}\left(\tau C_i + (1-\tau)\sum_{j=1}^{J} t_{ij}\right). \tag{11}$$

Considering delay and energy consumption, the total cost of local calculation is

$$C_{all} = \gamma t_{\text{all}} + (1-\gamma)c_{\text{all}}, \tag{12}$$

where γ is the weight.

The optimization problem can be formulated as

$$\min\left(C_{\text{all}}\right). \tag{13}$$

In order to ensure that the task is completed on time, the calculation task is required to complete the task before the vehicle leaves the MEC unit, and the following conditions shall be met:

$$t_i^{\text{stay}} = \frac{s_i}{v_i},$$
$$t + \frac{w_i}{e_{i,j}^{\text{mec}}} \le \min\left[t^{\max}, t_i^{\text{stay}}\right]. \tag{14}$$

Computing tasks unloaded to other servers should meet the following conditions:

$$2ct_w + t_{i,j}^{\text{trans}} + \frac{w_i}{e_{i,j}^{\text{mec}}} \le \min\left[t^{\max}, t_i^{\text{stay}}\right]. \tag{15}$$

The computing resources required to complete computing tasks are

$$e_{i,j}^{\text{mec}} \ge \max\left\{\frac{w_i}{\min\left[t^{\max}, t_i^{\text{stay}}\right] - t_{i,j}^{\text{trans}}}, \frac{w_i}{\min\left[t^{\max}, t_i^{\text{stay}}\right] - 2ct_w - t_{i,j}^{\text{trans}}}\right\}. \tag{16}$$

The total computing resources required for computing tasks are

$$e_j = \sum_{i=1}^{N}\sum_{x_i=j} e_{i,j}^{\text{mec}}. \tag{17}$$

The constraints are as follows:

$$C1\text{: } x_i \in \{0, 1, 2, \ldots, j\}, \forall i \in N,$$
$$C2\text{: } y_i \in \{0, 1\}, \forall i \in N,$$
$$C3\text{: } e_j < E_j, j \in \{1, 2, \ldots, J\}. \tag{18}$$

C1 indicates that a computing task can only be offloaded to one edge server and cannot be offloaded to two or more at the same time. C2 means that the computing task adopts binary offloading, which can choose not to offload or offload entire task at the same time, that is, the task is indivisible. C3 indicates that computing resources required by computing tasks offloaded to edge server cannot exceed the total resources of edge servers.

3. Solutions Based on Reinforcement Learning

3.1. Problem Solving Based on Distributed Reinforcement Learning. In view of dynamic, random, and time-varying nature of in-vehicle networks, artificial intelligence algorithms are more suitable for resource management and task scheduling than traditional mathematical methods. In comparison, Q-learning needs to maintain Q-table and is not suitable for networks with many states. Deep deterministic policy gradient algorithms need to use an experience replay mechanism to eliminate the correlation between training data. For experience playback mechanism, the agent consumes more resources for each interaction with the environment. The off-policy learning method adopted can only be updated based on the data generated by the old policy. Therefore, consider using the actor-critic algorithm to reduce the overhead required for algorithm execution, while providing optimal offloading decisions and resource management based on real-time network environment. Modeling the environment of system with an actor-critic algorithm requires determining its state space, action space, and reward function.

The state space, S, consists of computing resources and cache resources of in-vehicle network, $S = \{F_1, F_2, \ldots, F_M, S_1, S_2, \ldots, S_M\}$, where F_i and S_i represent the computing capacity and storage capacity of road side unit i, respectively.

The action space consists of offloading decisions of vehicles, caches of road side units, and computing resource management, $A = (x_i, w_i, f_i)$, where $x_i = \{x_{i0}, x_{i1}, \ldots, x_{iM}\}$, $w_i = \{w_{i1}, w_{i2} \ldots, w_{iM}\}$, and $f_i = \{f_{i1}, f_{i2} \ldots, f_{iM}\}$ represent the set of vehicle i offloading decision, road side unit storage, and computing resource management, respectively.

Reward Function. The goal of reinforcement learning training is to maximize long-term cumulative reward. According to the objective function of this paper, the reward function is designed as

$$r_{i,t} = 1 - \frac{C_{i,j}}{\max\{C_{i,j}\}}. \tag{19}$$

The public neural network in actor-critic algorithm consists of multiple threads, and each thread has the same 2 modules as public neural network: the policy (actor) network and the critic (critic) network. The actor network is used to optimize the policy $\pi(a_t|s_t;\delta)$ with parameters δ; the critic network tries to estimate the value function $V(s_t;\delta)$ with parameters δ_v. At time t, the actor network performs action a_t based on current state s_t, gets a reward r_t, and enters the next state s_{t+1}.

Use the advantage function $A(a_t, s_t)$ to represent the difference between the action value function $Q(a_t, s_t)$ and state value function $V(s_t)$:

$$A(a_t, s_t) = Q(a_t, s_t) - V(s_t). \tag{20}$$

To speed up convergence, approximate $Q(a_t, s_t)$ with k step sampling:

$$Q((a_t, s_t) \approx \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}; \delta_v), \tag{21}$$

where γ is the discount coefficient, r_{t+i} represents the instant reward, and $V(s_t)$ is obtained through critic network.

Taking the parameter δ as a variable, differentiate the policy loss function to obtain

$$\nabla_\delta f_\pi(\delta) = \nabla_\delta \log \pi(a_t||s_t;\delta) A(a_t, s_t) + \beta \nabla_\delta H(\pi(s_t;\delta)), \tag{22}$$

where H is the entropy of policy and β is the coefficient.

For the value loss function, there are

$$f_v(\delta_v) = (R_t - V(s_t;\delta))^2. \tag{23}$$

Based on RMSProp algorithm, the gradient estimate can be expressed as

$$g = ag + (1-a)\Delta\delta^2, \tag{24}$$

where a represents momentum and $\Delta\delta$ represents the cumulative gradient of loss function.

The update parameters of RMSProp algorithm are

$$\delta \leftarrow \delta - \eta \frac{\Delta\delta}{\sqrt{g+\varepsilon}}. \tag{25}$$

3.2. Algorithm Flow. The proposed offloading strategy flow based on distributed reinforcement learning is shown in Algorithm 1.

4. Example Verification and Result Discussion

4.1. Simulation Settings. This section uses Python to simulate and verify resource management optimization scheme for IoT and evaluate the pros and cons of different algorithms by comparing the impact of each algorithm on the total system overhead with the number of vehicles, the number of tasks, and the bandwidth. The simulation parameters are set as shown in Table 1. Due to the existence of small-scale fast fading and the mobility of mobile devices in established model, the results of each run are random. Therefore, the mathematical method of statistical averaging is used to obtain average value as the final result. The computer configuration information used for the simulation is Windows Server 2019, Intel(R) Xeon(R) 2.6 GHz processor, and 16 GB RAM.

4.2. Convergence Performance Analysis. Figure 2 describes the convergence of algorithm in this paper under different learning rate scenarios. It can be found from the figure that when the learning rates of actor and critical networks are L_a = 1×10−3 and L_b = 1×10−2, respectively, although the learning speed of algorithm is very fast, it will degrade the final convergence performance of system. When the learning rate is too small (L_a = 1×10−3, L_b = 1×10−2), the learning speed will drop sharply. Therefore, the learning rate is set to L_a = 1×10−4, L_b = 1×10−3 in the follow-up experiment.

4.3. Comparison of Accumulated Average Rewards under Different Schemes. Compare the average reward value of the

Input: actor network, actor target network, critical network and critical target network, learning rate α, discount rate γ, attenuation factor λ.
Output: computing task offloading policy π'.
Initialize the critical network parameter δ^c and actor network parameter δ^a.
Initialize the status of experience playback pool and task vehicle s_0
For $t \leq T$ **do**
Observe the environment status s_t and select actions a_t based on the current policy
Execute the action a_t, get the reward r_t, and transfer to the state s_{t+1}
Save array (s_t, a_t, r_t, s_{t+1}) to experience playback pool
If the memory bank is full, but the stop condition is not met, a small batch of arrays (s, a, r, s') is randomly sampled from the experience playback pool.
Update critical network parameters, actor network parameters, and target network parameters
End
End

Algorithm 1: Resource management algorithm based on distributed reinforcement learning.

Table 1: System parameters.

Parameter	Value
Number of concurrent tasks of a single vehicle	2~7
Number of vehicles	5~15
Calculation capability of onboard terminal	5 MHz
Vehicle transmission power	1.5 W
Vehicle speed	30–80 KM/h
Gaussian white noise power	-80 dB
MEC computing power	$[2\times10^8, 9\times10^8]$ Hz
System bandwidth	10~50 MHz

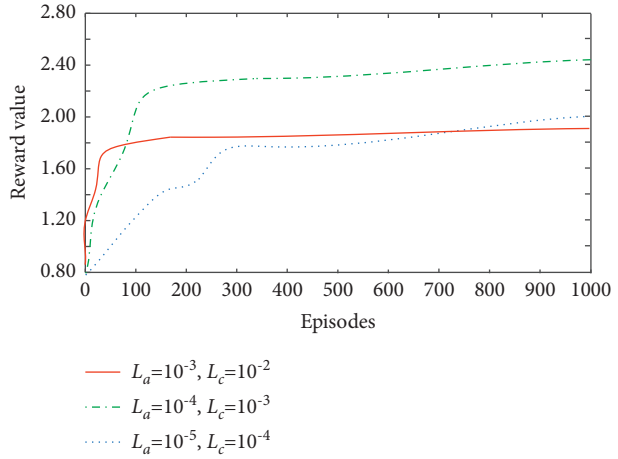

Figure 2: Performance comparison under different learning rates.

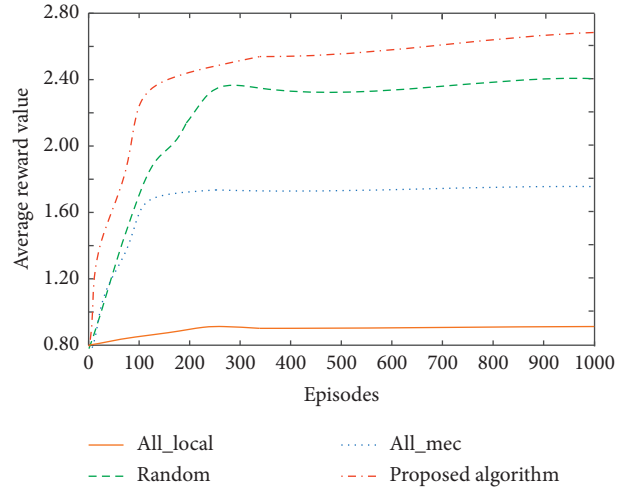

Figure 3: Comparison of average rewards under different schemes.

proposed scheme with the following schemes: (1) all-local strategy; (2) random strategy; (3) all-MEC policy. During DDPG training, there will be violent shocks. Thus, this section observes the convergence of neural network by calculating the cumulative average value of system reward. Figure 3 shows the comparison of cumulative average rewards for different caching schemes. With the increase of training times, it can be seen that all-MEC and random schemes can gradually converge to a stable cumulative average. All-local strategy behavior is not encouraged, so the reward value is the lowest. Because the proposed algorithm needs to consider the road conditions of adjacent areas, increases the dimension of system state, and improves the complexity, it has poor performance at the beginning of training and obtains the highest average reward value after convergence. Therefore, the proposed resource optimization management scheme for IoT can make full use of communication resources and effectively improve the effectiveness of the system.

4.4. Performance Comparison under Different Algorithms. In order to prove the advantages of the proposed algorithm, the algorithms in [28–29] are compared with the proposed algorithm under the same experimental conditions. Figure 4 shows the impact of the number of vehicles on the delay. It can be found that the delay of system task processing increases with the increase of the number of vehicles. This is mainly due to the increase of processing tasks and the limitation of computing resources. Among all algorithms, The reference [29] algorithm has the largest delay.

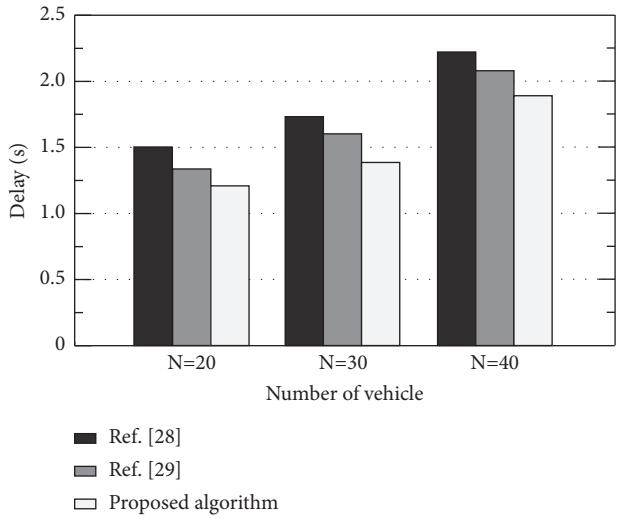

Figure 4: Influence of vehicle number on delay of different algorithms.

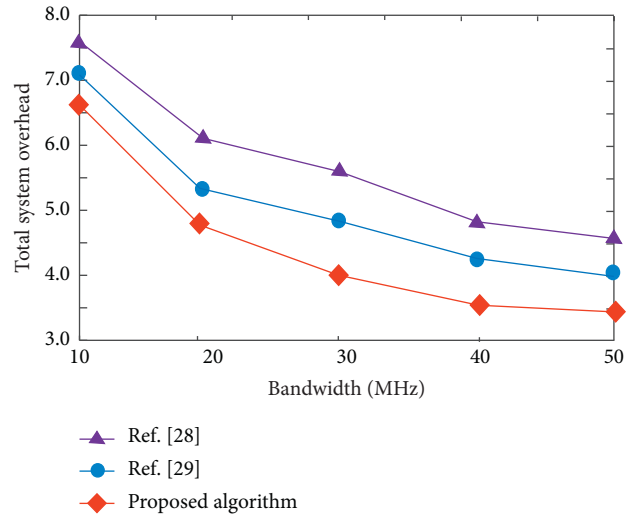

Figure 5: Variation of total system overhead with bandwidth under different algorithms.

Compared with the reference [29] algorithm and the proposed algorithm, the vehicle will undertake more tasks. Due to the limitation of the vehicle's own computing resources, processing tasks alone will cause greater delay. Due to the limitation of vehicle computing resources, processing tasks alone will cause large time delay. The proposed algorithm considers the cooperation of terminal, edge, and cloud, improves the utilization efficiency of resources, and minimizes the system delay.

The change of total system overhead with bandwidth under different algorithms is shown in Figure 5. Figure 5 shows that with the increase of bandwidth, the total system overhead of three algorithms shows a downward trend, but the total system overhead of the proposed algorithm is always lower than that of other two algorithms. When bandwidth = 40 MHz, the total system overhead of the algorithm in [28] is 4.732, and the total system overhead of the algorithm in [28] algorithm is 4.251, while the total system overhead of the proposed algorithm is only 3.502. Further analysis shows that when the cloud computing ability of comparison algorithm is relatively weak, most of computing tasks will be completed at the edge node, which cannot make good use of the cloud edge system. Therefore, it produces high total system overhead. Compared with the other two algorithms, the proposed algorithm can achieve the lowest total system overhead because the proposed algorithm considers the dynamic, random, and time-varying characteristics of vehicle network to optimize system performance to the greatest extent.

The change of total system overhead with the number of tasks is shown in Figure 6. With the increase of the number of tasks, the total cost of the three algorithms shows an upward trend. However, the total system overhead of this algorithm is less than that of the algorithms in [28, 29]. This is because the algorithm can collect the state and action information of the whole system and make better decisions according to the global information, so the total cost of the system is low. The comparison algorithm does not fully analyze the state and action information of the system, and

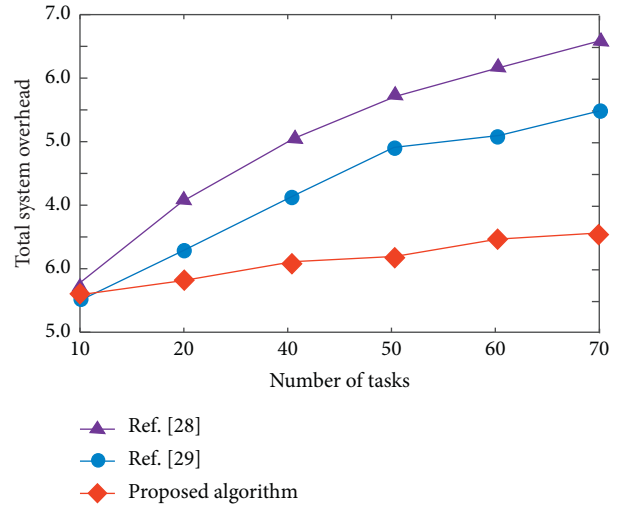

Figure 6: Changes of total system overhead with the number of tasks under different algorithms.

the multi-vehicle game increases the energy consumption, resulting in the increase of the total cost of the system.

5. Conclusion

Aiming at the problem of delay and energy consumption caused by the contradiction between limited network resources and a large number of user needs in vehicle environment, this paper proposes an efficient resource management optimization scheme for IoT in edge computing environment. The proposed algorithm builds a communication model and task offloading optimization model in IoT edge computing scenario and solves them based on distributed reinforcement learning to maximize the system performance.

In the future, we will study the content acquisition decision combined with micro-traffic data and the

prediction of vehicle mobility to further improve algorithm performance. Besides, a dynamic situation will be considered, that is, devices may leave the current edge server during computing offloading. In this case, it is necessary to set up a more effective mobile model for devices. In addition, the current blockchain technology provides a powerful solution for the unloading of secure computing tasks in the IoV. In view of the contradiction between the high real time of IoV applications and the low real time of blockchain, we can take advantage of the differences of participants in IoV in terms of security level, computing power, and communication ability to design a hierarchical blockchain structure matching the cloud IoV structure to solve it. It is of great significance to use the blockchain technology to build a secure Internet of Vehicles computing task unloading platform.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] D. Zhang, L. Cao, H. Zhu, T. Zhang, J. Du, and K. Jiang, "Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning," *Cluster Computing*, vol. 25, no. 2, pp. 1175–1187, 2022.

[2] M. S. Aslanpour, A. N. Toosi, and C. Cicconetti, "Serverless edge computing: vision and challenges," in *Proceedings of the 2021 Australasian Computer Science Week Multiconference*, pp. 1–10, Dunedin, New Zealand, February, 2021.

[3] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," *IEEE Access*, vol. 8, no. 2, Article ID 85714, 2020.

[4] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[5] N. Hassan, K.-L. A. Yau, and C. Wu, "Edge computing in 5G: a review," *IEEE Access*, vol. 7, no. 6, Article ID 127276, 2019.

[6] D. Zhang, N. Vance, and D. Wang, "When social sensing meets edge computing: vision and challenges," in *Proceedings of the 2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9, IEEE, Valencia, Spain, July, 2019.

[7] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[8] Z. Fan, W. Yang, and K. Tian, "An edge computing service model based on information-centric networking," in *Proceedings of the 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 498–505, IEEE, Tianjin, China, December, 2019.

[9] S. Chen, Q. Li, M. Zhou, and A. Abusorrah, "Recent advances in collaborative scheduling of computing tasks in an edge computing paradigm," *Sensors*, vol. 21, no. 3, pp. 779–787, 2021.

[10] W. Wu, Q. Zhang, and H. J. Wang, "Edge computing security protection from the perspective of classified protection of cybersecurity," in *Proceedings of the 2019 6th International Conference on Information Science and Control Engineering (ICISCE)*, pp. 278–281, IEEE, Shanghai, China, December, 2019.

[11] S. M. Khan, M. Chowdhury, E. A. Morris, and L. Deka, "Synergizing roadway infrastructure investment with digital infrastructure for infrastructure-based connected vehicle applications: review of current status and future directions," *Journal of Infrastructure Systems*, vol. 25, no. 4, Article ID 03119001, 2019.

[12] F. Loussaief, H. Marouane, H. Koubaa, and F. Zarai, "Radio resource management for vehicular communication via cellular device to device links: review and challenges," *Telecommunication Systems*, vol. 73, no. 4, pp. 607–635, 2020.

[13] A. W. Wang, Y. J. Wang, A. M. Zahm, A. R. Morgan, K. J. Wangensteen, and K. H. Kaestner, "The dynamic chromatin architecture of the regenerating liver," *Cellular and molecular gastroenterology and hepatology*, vol. 9, no. 1, pp. 121–143, 2020.

[14] R. Shrestha, S. Y. Nam, R. Bajracharya, and S. Kim, "Evolution of V2X communication and integration of blockchain for security enhancements," *Electronics*, vol. 9, no. 9, pp. 1338–1345, 2020.

[15] X. Li, L. Zhou, Y. Sun, S. Zhou, and M. Lu, "Resource allocation schemes based on improved beetle antennae search algorithm for collaborative communication of the unmanned aerial vehicle network," in *Proceedings of the International Conference on Wireless and Satellite Systems*, pp. 275–282, Springer, Harbin, China, January, 2019.

[16] T. Xue, W. Wu, Q. Wang, and X. Wu, "Radio resource allocation for V2X communications based on hybrid multiple access technology," in *Proceedings of the International Conference on Wireless and Satellite Systems*, pp. 23–35, Springer, Harbin, China, January, 2019.

[17] C. He, Q. Chen, C. Pan, X. Li, and F.-C. Zheng, "Resource allocation schemes based on coalition games for vehicular communications," *IEEE Communications Letters*, vol. 23, no. 12, pp. 2340–2343, 2019.

[18] J. Kim, J. Lee, S. Moon, and I. Hwang, "A position-based resource allocation scheme for V2V communication," *Wireless Personal Communications*, vol. 98, no. 1, pp. 1569–1586, 2018.

[19] F. Wang, J. Liu, L. Zhao, and K. Zheng, "Decentralised resource allocation of position-based and full-duplex-based all-to-all broadcasting," *IET Communications*, vol. 13, no. 15, pp. 2254–2260, 2019.

[20] M. Chen, T. Wang, K. Ota, M. Dong, M. Zhao, and A. Liu, "Intelligent resource allocation management for vehicles network: an A3C learning approach," *Computer Communications*, vol. 151, no. 1, pp. 485–494, 2020.

[21] Y. Liu, H. Yu, S. Xie, and Y. Zhang, "Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 11, Article ID 11158, 2019.

[22] Y. Zhan, S. Guo, P. Li, and J. Zhang, "A deep reinforcement learning based offloading game in edge computing," *IEEE Transactions on Computers*, vol. 69, no. 6, pp. 883–893, 2020.

[23] L. Huang, S. Bi, and Y.-J. A. Zhang, "Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks," *IEEE Transactions on Mobile Computing*, vol. 19, no. 11, pp. 2581–2593, 2020.

[24] Y. Hui, Z. Su, T. H. Luan, and J. Cai, "Content in motion: an edge computing based relay scheme for content dissemination in urban vehicular networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 8, pp. 3115–3128, 2019.

[25] Z. Su, Y. Hui, and S. Guo, "D2D-based content delivery with parked vehicles in vehicular social networks," *IEEE Wireless Communications*, vol. 23, no. 4, pp. 90–95, 2016.

[26] W. Zhao, Y. Qin, D. Gao, C. H. Foh, and H.-C. Chao, "An efficient cache strategy in information centric networking vehicle-to-vehicle scenario," *IEEE Access*, vol. 5, no. 9, pp. 12657–12667, 2017.

[27] Y. Zhang, T. M. Mirpuri, and C. L. Ho, "Primary epithelioid sarcoma manifesting as a fungating scalp mass - imaging features and treatment options. A case report and literature review," *Journal of Radiology Case Reports*, vol. 15, no. 11, pp. 1–9, 2021.

[28] C. Zhang, H. Du, and Q. Gao, "DMORA: decentralized multi-SP online resource allocation scheme for mobile edge computing," *IEEE Transactions on Cloud Computing*, vol. 9, no. 1, p. 1, 2020.

[29] X. Li, "A computing offloading resource allocation scheme using deep reinforcement learning in mobile edge computing systems," *Journal of Grid Computing*, vol. 19, no. 3, pp. 2001–2009, 2021.