Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 6174708, 8 pages
https://doi.org/10.1155/2022/6174708

Research Article

5G Converged Network Resource Allocation Strategy Based on Reinforcement Learning in Edge Cloud Computing Environment

Xuezhu Li

Department of Information Engineering, Suzhou University, Suzhou, Anhui 234000, China

Correspondence should be addressed to Xuezhu Li; lixuezhu019@163.com

Received 21 January 2022; Revised 23 March 2022; Accepted 25 April 2022; Published 14 May 2022

Academic Editor: Laith Abualigah

Aiming at the problem that computing power and resources of Mobile Edge Computing (MEC) servers are difficult to process long-period intensive task data, this study proposes a 5G converged network resource allocation strategy based on reinforcement learning in edge cloud computing environment. *n* order to solve the problem of insufficient local computing power, the proposed strategy offloads some tasks to the edge of network. Firstly, we build a multi-MEC server and multi-user mobile edge system, and design optimization objectives to minimize the average response time of system tasks and total energy consumption. Then, task offloading and resource allocation process is modeled as Markov decision process. Furthermore, the deep Q-network is used to find the optimal resource allocation scheme. Finally, the proposed strategy is analyzed experimentally based on TensorFlow learning framework. Experimental results show that when the number of users is 110, final energy consumption is about 2500 J, which effectively reduces task delay and improves the utilization of resources.

1. Introduction

With the continuous development of technologies such as the 5G, the amount of data in various emerging application scenarios has exponentially increased. There are more and more Internet of Things (IoT) devices in various fields such as telemedicine, smart car driving, and smart cities, so all kinds of computing are everywhere [1]. However, the existing cloud computing models are difficult to manage these large-scale computing resources and perform data analysis. This is mainly reflected in the following two reasons: First, the transfer of large-scale data to cloud computing center will improve network performance and the computing power of cloud computing infrastructure brings severe challenges [2,3]. The second is that it is difficult for cloud far away from users to meet the stringent requirements of new applications such as autonomous driving on network delay and response speed [4]. Thus, both computing services and big data sources are undergoing a shift from cloud to edge [5].

Edge computing serves as an intermediate layer between the cloud computing center and user devices. It provides computing resources to users near the edge via a high-speed network by placing the edge server close to user end [6]. Among them, the user device sends computing tasks that originally need to be sent to the cloud or executed locally to edge server for execution to achieve reasonable network resource allocation, which is called computing offloading [7]. Compared with cloud servers and local computing, edge computing can provide faster network response and have more powerful computing capabilities [8]. Therefore, computing offloading and reasonable allocation of network resources by a reasonable scheduling algorithm can help users save transmission energy consumption and improve computing efficiency [9].

In the edge computing system, for security and efficiency reasons, the edge server will not open its own computing resource configuration and idle state to each user device, so it is difficult to obtain the detailed status in the system [10,11]. Under the premise of incomplete observation system constraints, task offloading and system optimization problems become more complicated. The intelligent model represented by deep reinforcement learning is an important means to solve such problems [12]. Reference [13] developed a Multi-Agent Reinforcement Learning Network to solve the Q-learning problem based on independent learners, and

designed the calculation unloading strategy in IoT through random game. However, the efficiency of resource allocation strategy needs to be further improved. Reference [14] proposed a moving edge computing (MEC) network based on blockchain, which uses blockchain to control the coverage system, and adopts adaptive strategy to generate blocks and realize high-quality resource allocation. Reference [15] used deep Q network (DQN) learning to obtain the best resource allocation scheme in IoT network. However, frequent data interaction brings high network load, which becomes the main obstacle to the training of intelligent offloading models, especially computing offloading models based on deep learning.

Traditional methods also have certain research on computing task offloading and network resource allocation: For example, reference [16] solved the task unloading problem based on differential evolution algorithm, so as to realize the efficient execution of tasks, but it requires higher network bandwidth. Reference [17] designed a random mixed integer nonlinear programming method for the intensive task unloading and resource allocation in MEC, which can realize the rational use of resources, but cannot take into account energy efficiency and service delay. Reference [18] used orthogonal and non-orthogonal multiple access methods, a resource allocation scheme considering energy consumption and efficiency in MEC is formulated, but the overall delay needs to be further reduced. Reference [19] proposed a multi-objective resource allocation method for MEC, which uses Pareto archiving evolution strategy to optimize time cost and load balancing. At the same time, it combined multi-criteria decision-making and sorting preference technology similar to the ideal solution to obtain optimal resource allocation, but 5G integration scheme is not considered.

Aiming at the problem that the large amount of data transmitted in 5G network leads to channel congestion, which affects the real-time performance and energy consumption of communication, a 5G fusion network resource allocation strategy based on reinforcement learning in edge cloud computing environment is proposed. Due to the poor learning effect of basic reinforcement learning in massive data, the proposed strategy proposes a DQN offloading strategy to solve resource allocation of 5G converged networks, which can reduce the time delay. At the same time, the system energy consumption is reduced. Finally, experimental results based on TensorFlow learning framework show that proposed strategy fully considers the time and energy consumption of local and offloading to MEC execution, and solving offloading scheme by reinforcement learning can greatly reduce delay and energy consumption. Moreover, its energy consumption is about 2500J, the time delay does not exceed 7s. DQN has self-learning ability, which continuously learns during the training process to improve the accuracy of decision-making. Therefore, it can effectively reduce load and broadband utilization rate.

2. System Scenario and Optimization Goal

2.1. System Scenario. The system scenario is shown in Figure 1, consisting of N users, M base stations, and multiple MEC servers. Among them, each user is associated with the nearest base station through the wireless link and sends a task request to it. At the same time, each base station is equipped with an MEC server with multiple CPU cores. Therefore, MEC server can process the computing tasks of different users in parallel. It is assumed that user computing tasks are processed by an MEC server, regardless of situation in which computing tasks are forwarded between MEC servers.

Divide the system running time dimension into a number of time slots, and use $T = \{0, 1, 2 \cdots\}$ to represent the set of time slots for network operation, where the time length of each time slot t is defined as τ. It is assumed that most of the computing tasks of user can be processed and completed in one time slot. Due to the large amount of data, some computing tasks are divided into subtasks for processing [20]. Considering the randomness of task arrival, a two-level queue model is designed to describe the state of computing tasks, namely the user task queue model and MEC server task queue model.

2.2. Task Generation Model. In MEC model, it is assumed that the time interval for mobile users to generate tasks obeys Poisson distribution, and user n generates k_n mutually independent tasks, which are defined as $K_n = \{1, 2, \cdots, k_n\}$. The attribute of task i is defined as $\varphi_i = \{id_n, id_i, sub_i, d_i, c_i, l_i, mem_i, cpu_i\}$, where id_u represents the identity (id) of user n who generated task i, id_i represents the id of task, and sub_i represents the time when the user submits the task. d_i (bits as a unit) represents the amount of task data, c_i (CPU revolutions/bit as a unit) represents the number of CPU revolutions required to calculate one bit of task data, and $l_i = d_i c_i$. mem_i and cpu_i, respectively represent the memory and CPU resources required by computing tasks. Users are mobile and may be located near different base stations at different points in time. Thus, tasks generated by same user may be offloaded to servers in different base stations for processing.

2.3. Calculation Model

2.3.1. Local Calculation Model. Mobile users themselves have certain computing capabilities. If the user has sufficient computing resources, then tasks can be processed locally. The computing power of local device n is represented by CPU frequency, which is defined as $f_{n,l}$. The processing time of task on local calculation model only considers the calculation time. Therefore, the local processing time of task i generated by user n is defined as

$$t_{i,n} = \frac{l_i}{f_{n,l}}. \tag{1}$$

The power and energy consumption of task i processed locally by user n are, respectively, defined as

$$\begin{cases} P_{i,n} = \gamma\left(f_{n,l}\right)^3 \\ E_{i,n} = \gamma\left(f_{n,l}\right)^2 l_i \end{cases}, \tag{2}$$

where γ is the effective switch capacitance.

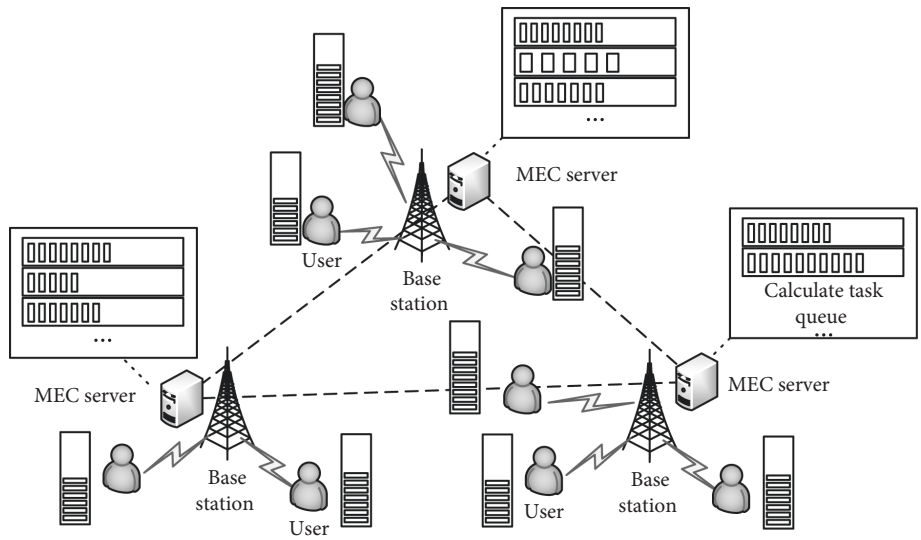

Figure 1: System scenario.

2.3.2. Edge Computing Model. Due to the insufficient computing resources of local devices, a large number of tasks generated by users cannot be processed on local computing model, but some tasks need to be offloaded to edge computing model for processing [21]. When the task is executed on MEC, the transmission time and calculation time need to be considered, and the amount of data returned by the task is very small, so the transmission time does not consider the time-consuming of result return. Before calculating the transmission time, first define the transmission rate from user device n to base station m as

$$v_{n,m} = B \log_2\left(1 + \frac{p_n h_{n,m}}{\delta_0 B}\right), \tag{3}$$

where B is the communication bandwidth; p_n is the transmission power of user n. δ_0 is the noise power spectral density of base station m; $h_{n,m}$ is the channel gain between user n and base station m.

The he time for task i generated by user device n to be offloaded to server j of base station m for processing is defined as

$$t_{i,off} = \frac{d_i}{v_{n,m}} + \frac{l_i}{f_{m_j}}, \tag{4}$$

where f_{m_j} is the CPU frequency of server j on base station m.

Same as time-consuming calculation, the energy consumption of task i generated by user n and offloaded to server j of base station m for processing is defined as

$$E_{i,off} = p_n \frac{d_i}{v_{n,m}} + d_i e_{m_j}, \tag{5}$$

where e_{m_j} is the energy consumption required to calculate one bit of data.

2.4. Optimization Goal. The optimization goal is to reduce the average response time and total energy consumption of tasks in MEC environment, improve user service quality and save system energy cost [22]. The execution of tasks on computing nodes will be constrained by some network hardware environments [23]. Suppose that the maximum number of tasks that can be executed in parallel on the computing node is Γ, if the number of tasks is less than Γ, new tasks can be received; Otherwise, you need to wait for the resource release task to be executed. In addition, a new task can be processed only when the network resource required by the executing task and the new task is less than the total resources. The mathematical expression of the objective optimization problem is as follows:

$$\begin{cases} \min\limits_{\{q\}} \overline{T} \text{ and } E_{total}\overline{T} = \sum\limits_{k=1}^{K} \frac{\widehat{T}_i}{k}, \widehat{T}_i = T_i^{\infty} - sub_i \\ E_{total} = \sum\limits_{i=1}^{k_n} E_i \end{cases},$$

$$s.t.\ \ q \le \Gamma, \tag{6}$$

$$\sum_{i=1}^{q} mem_i \le C_1,$$

$$\sum_{i=1}^{q} cpu_i \le C_2,$$

where q is the number of simultaneous tasks, and C_1 and C_1 are memory and CPU capacity respectively, and T_i^{∞} is the completion time of task i, $E_i = \begin{cases} E_{i,n}, local\ computing \\ E_{i,off}, offloading\ computing \end{cases}$.

3. Solutions Based on Deep Reinforcement Learning

3.1. Markov Decision Process Modeling. The Markov decision process is described by quadruple $\langle S, A, \psi, R \rangle$:

(1) S is the system state collection. For an incomplete observation system, the set used by edge server to describe the system state only includes the basic information of edge server: $S = \{S_1, S_2, \cdots, S_J\}$. Among them, let S_J be a 5-tuple.

(2) $a_n^t \in A$ is a finite set of actions, that is, the action of calculating offloading. The set includes the user who decides to uninstall at time t, and the user's action at time t is recorded as a_n^t. When $a_n^t = 0$, user n executes locally. When $a_n^t = 1$, user n offloads the task to the MEC.

(3) ψ is the state transition matrix, and ψ corresponds to the mapping of $S \times A \times S \longrightarrow [0, 1]$. That is, the probability of transitioning to next state after the end of state **S**, after the execution of action **A**.

(4) R is the reward function. When the user needs to uninstall, the uninstall action will get a positive reward. When the decision makes system overload, a negative reward, or penalty, is given.

Reinforcement learning obtains rewards through reward function r_t at time t. For some observable system environments, the remote server can only obtain information about tasks that have been offloaded to the remote [24,25]. Therefore, it is considered that the amount of calculation saved is regarded as a reward for an offloading behavior. In order to better master the use of system resources, a punitive reward will be set. The punitive reward is set to the negative value of absolute value of current system reward, which ensures that the punitive reward value is always negative [26,27]. The punitive reward is expressed as

$$\overline{r}_t = -\left|r_{t-1}\right| \tag{7}$$

Markov process corresponds to a sequence of system state transitions, that is, a trajectory sequence $\Xi = \langle s^0, a^0, s^1, a^1, \cdots \rangle$ containing states and actions can be obtained. Strategy π corresponds to the mapping of $S \times A \longrightarrow [0, 1]$. Deep reinforcement learning maximizes the cumulative reward expectation of Ξ during the training process to find optimal strategy π.

3.2. DQN-Based Offload Strategy. The training process based on DQN offloading strategy is shown in Figure 2.

According to the above figure, the pseudo code of algorithm based on DQN offloading strategy is shown in Algorithm 1.

Based on DQN algorithm, two neural network structures, the current Q-value network and target Q-value network are used. The two have the same neural network structure, but the parameters of their respective structures are different. The definition θ represents the parameters of current Q-value network, and θ' represents the parameters of target Q-value network. DQN algorithm fits the action value function $Q(s_t, a_t; \theta)$ through Q-value network with parameter θ, which is calculated as follows:

$$Q(s_t, a_t; \theta) = E\left[\sum_t \chi_t R(s_t, a_t) | s, a\right], \tag{8}$$

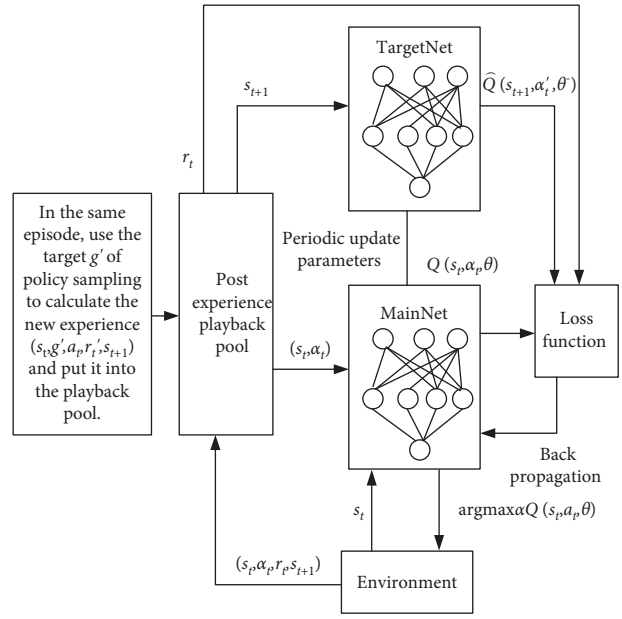

Figure 2: Training process offloading strategy based on DQN.

where $\chi \in [0, 1]$ is the reward discount factor.

Then select the optimal action based on value of each action generated by Q-value network:

$$a_t = \arg\max_a Q(s, a; \theta). \tag{9}$$

In order to avoid not selecting the optimal local optimal solution when selecting an action, ε-greedy strategy is used to select an action. That is, an action is randomly selected with a small probability of ε, and the optimal action is selected according to (10) with a probability of $1-\varepsilon$, so as to obtain the reward value r_t and next state s_{t+1}. Then put quadruple (s_t, a_t, r_t, s_{t+1}) into the experience replay library, and sample a batch of (s_t, a_t, r_t, s_{t+1}) into the neural network for training. When action a_t is executed, the Q-value corresponding to action space a_t is updated, according to Bellman formula:

$$Q(s, a; \theta) = R(s, a) + \chi \max_{a'} Q(s_{t+1}, a'; \theta'). \tag{10}$$

Then minimize Loss function to update the parameters of current Q-value network. Loss function represents the predicted value of square error loss between the current Q-value network and target Q-value network. The smaller the value, the better the neural network is optimized. Generally expressed as

$$L(\theta) \approx E\left\lfloor \left(r_t + \chi \max_{a'} Q(s_{t+1}, a'; \theta') - Q(s_t, a_t; \theta) \right)^2 \right\rfloor. \tag{11}$$

Then the target Q-value network is updated with a delay.

4. Experiment and Analysis

The platform used in the experiment is Python 3.6, and Tensorflow GPU 1.14 is used for in-depth learning and

Input: Target resampling strategy S, Reward function R: $S \times A \times g \longrightarrow R$
Begin
(1) Initialize replay pool
(2) For episode = 0, 1, 2, . . ., m
do Initialize a state s_0 and a target g;
(3) For $t = 0, 1, 2, \ldots, T-1$
doUse behavior strategies to select actions a_t
Execute action a_t and observe the new state s_{t+1}
(4) End for
(5) For $t = 0, 1,2, \ldots, T-1$
do Calculate immediate rewards
Put (s_t, g, a_t, r_t, s_t) this experience is stored in the playback pool
Resample a batch of target G using the target resampling policy **S**
(6) For $g' \in G$
doCalculate new immediate rewards r'
Put $(s_t, g', a_t, r'_t, s_{t+1})$ this new experience is stored in the playback pool
(7) End for
(8) End for
(9) For $t = 0,1,2, \ldots,$ N
do Sample some minibatch from the replay pool
Calculate the loss function and update the network parameters
(10) End for
(11) End for
End

Algorithm 1: Pseudo code of offloading strategy based on DQN.

optimization training. Meanwhile, the simulation parameter settings are shown in Table 1.

In addition, the proposed strategy is compared with reference [13], reference [18], and reference [19] to demonstrate its advantages. Among them, reference [13] proposed a Multi-Agent Reinforcement Learning Algorithm for computing offload of Internet of things edge computing network; Reference [18] formulated a resource allocation strategy based on orthogonal and non-orthogonal multiple access schemes; Reference [19] uses Pareto archive evolution strategy to achieve multi-objective resource allocation.

Table 1: Experimental parameter and hyperparameter setting.

Parameter	Value
c_i (Cycles/bit)	500
$f_{n,i}$ (GHz)	Unif (0.5.1.0)
$f_{m,j}$ (GHz)	Unif (5.0,10.0)
p_n (mW)	150
C (%)	150
$D_{n,m}$ (m)	randint (50,200)
C_1 (GB)	32
δ_0 (dBm/Hz)	−175
Ω	12
χ	0.7

4.1. Analysis of Energy Consumption Results. The relationship between the number of users and energy consumption for the four strategies is shown in Figure 3.

It can be seen from Figure 3 that the energy consumption of each strategy basically shows an upward trend. However, the rise of proposed strategy has slowed down. When the number of users is 110, the final energy consumption is about 2500J. This is because too many users lead to full load of edge computing nodes, so tasks are offloaded to higher-performance cloud data centers, keeping the energy consumption of proposed strategy to a low level. Besides, it comprehensively considers local and offloading energy consumption using DQN to obtain the optimal offloading plan, which can effectively reduce energy consumption. In reference [13], although multi-agent reinforcement learning algorithm is used to obtain the optimal offloading plan, the cloud data center is not considered, so the energy consumption is increasing rapidly. The other two strategies are difficult to handle increased number of users, and the energy consumption is higher, exceeding 3500J.

4.2. Analysis of Time Delay Results. Similarly, the relationship between users and time delay under the four strategies is shown in Figure 4.

It can be seen from Figure 4 that reference [18] preferentially chooses to execute tasks locally to meet the requirements of delay-sensitive tasks. If the computing resources are insufficient, it will turn to high-level devices for offloading, so the delay is almost the lowest, no more than 5s. However, the strategy in reference [19] tends to preferentially offload tasks to edge nodes, and the increase in the number of users will reasonably uninstall some tasks, because the computing resources of edge nodes are in short supply and need to be queued for use, the delay will increase suddenly. As the number of users further increases, tasks are reasonably offloaded, which can alleviate time delay to a certain extent. But because of transmission link, although there is no need to queue up, a lot of time is lost in the transmission process. Even if the task continues to increase, time delay will stabilize in a higher range, about 17s.

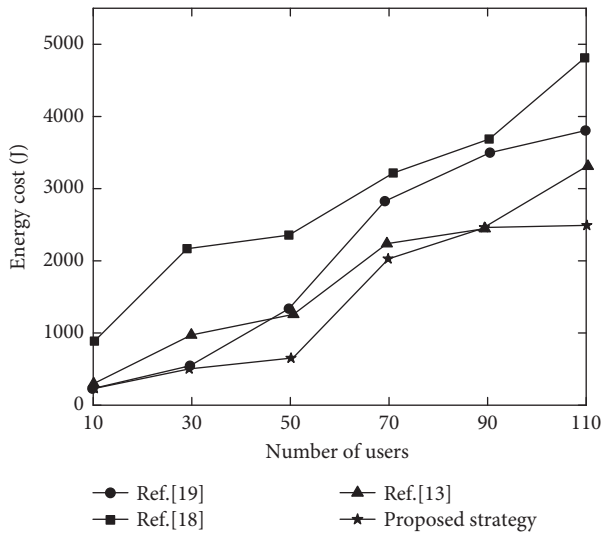

Figure 3: Comparison of energy consumption of different strategies.

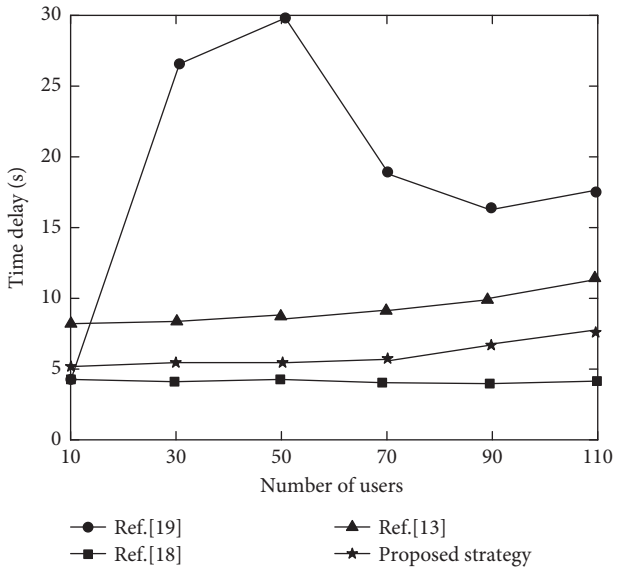

Figure 4: Comparison of time delay of different strategies.

However, the delays of reference [13] and proposed strategy are relatively stable. The proposed strategy fully considers the time and energy consumption of local and offloading to MEC execution, and solving offloading scheme by reinforcement learning can greatly reduce delay.

4.3. Analysis of Load Balancing Rate Results. Figure 5 shows the relationship between users and load balancing ratios under the four strategies.

It can be seen from Figure 5 that the overall load balancing ratio of reference [18] strategy is relatively high. This is because it focuses on local execution, and task offloading starts from the device with the lowest performance, so as long as the device performing the task is almost fully loaded. Although some pressure was shared between 30 and 70 by offloading to edge nodes, the resources of edge nodes were

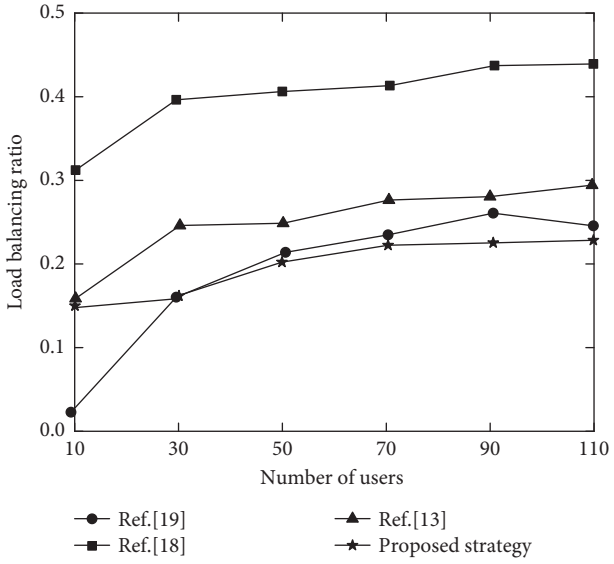

Figure 5: Load balancing rate of different strategies.

quickly occupied. However, the strategy in reference [19] tends to be offloaded to MEC server, so the load balancing rate is low. This can maintain a high utilization rate for a relatively large number of edge node clusters with moderate performance. Reference [13] used multi-agent reinforcement learning algorithms for task offloading, but the load balancing rate is low. However, the algorithm performance still needs to be improved compared with DQN, so the load balancing rate of proposed strategy is the lowest, about 0.23. Reasonable utilization of users, MEC and cloud center can greatly reduce the load balancing rate.

4.4. Impact of Different Similarity Measurement Methods on Algorithm Execution Efficiency. According to the pipeline model, bandwidth resource bottleneck is the first dilemma faced in the offloading process. Ensuring the effective use of bandwidth resources, rather than blindly offloading too many tasks, is the key to rational use of system resources. With the increase in the number of users, the four strategies are shown in Figure 6 for network and server usage.

It can be seen from Figure 6 that compared with other strategies, the broadband utilization rate and computing resource utilization rate of proposed strategy is relatively low. Among them, the broadband utilization rate is always between 0.1 and 0.3, and computing resource utilization rate is roughly between 0.2 and 0.45. Since the proposed strategy always occupies a lower bandwidth in the decision-making process, DQN strategy is used to reasonably offload computing tasks, thereby avoiding bottlenecks in network transmission. At the same time, because fewer broadband resources are occupied, higher revenue can be obtained for servers. Reference [13] performed computational offloading based on multi-agent reinforcement learning algorithm. Although the task offloading can be completed well, MEC server has a higher requirement for computing power, so it occupies more computing resources. Reference [18] and reference [19] lacked high-performance processing algorithms and cannot balance

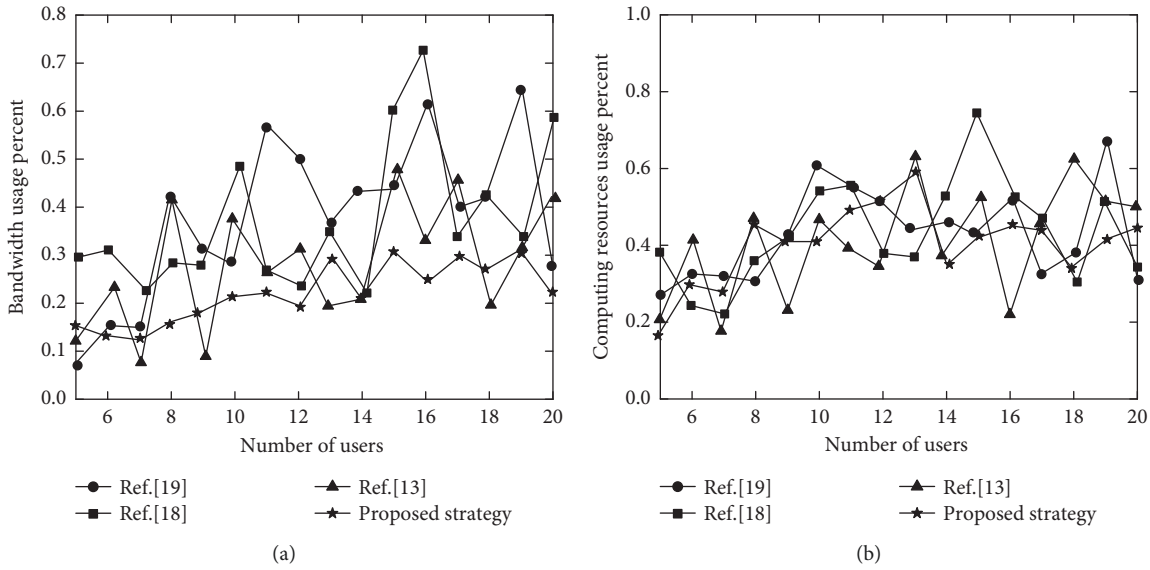

Figure 6: Changes of network and server load with the number of users. (a) Bandwidth usage percent. (b) Computing resources usage percent.

task offloading. Thus, the broadband utilization rate and computing resource utilization rate fluctuate greatly and are at a high value.

5. Conclusion

With the rapid development of IoT and 5G technology, a series of new applications with computationally intensive and delay-sensitive features such as virtual reality, augmented reality and face recognition continue to emerge. In order to solve the problem of insufficient local computing power, the proposed strategy offloads some tasks to the edge of network, and builds a mobile edge system model with multi-MEC server and multi-user. This model improves the task processing capability of system by solving goal of minimizing. Besides, DQN strategy is used to obtain an offloading plan that minimizes the average response time of system tasks and total energy consumption, so as to allocate computing resources reasonably.

The proposed strategy has certain value and significance for theoretical research and practical application. However, due to resource constraints such as mobile devices, servers and base stations, experiments can only be carried out in a simulated environment that is as close to the actual situation as possible. In the future research work, we will further consider conducting physical experiments in a real environment to provide solutions to practical problems.

Data Availability

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

Conflicts of Interest

The authors declare that there are conflicts of interest regarding the publication of this study.

Acknowledgments

This work was supported by The 2019 Anhui Province University Outstanding Top Talent Cultivation Funding Project, "Application Research of Task Scheduling Based on Energy Awareness in Heterogeneous Hybrid Cloud Environment" (Item Number: gxgnfx2019050) and The Second Batch of Industry-University Cooperation Collaborative Education Projects in 2020, Blockchain + big data professional teacher training and course construction for new engineering (Item Number: 202101207023), Demonstration Project of Provincial Grassroots Teaching and Research Office of Software Engineering: (Item Number: 2020SJSFJXZZ417).

References

[1] W. Wen, Y. Cui, T. Q. S. Quek, F. C. Zheng, and S. Jin, "Joint optimal software caching, computation offloading and communications resource allocation for mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7879–7894, 2020.

[2] T. Alfakih, M. M. Hassan, A. Gumaei, C. Savaglio, and G. Fortino, "Task offloading and resource allocation for mobile edge computing by deep reinforcement learning based on SARSA," *IEEE Access*, vol. 8, no. 6, pp. 54074–54084, 2020.

[3] H. Zhang, Z. Wang, and K. Liu, "V2X offloading and resource allocation in SDN-assisted MEC-based vehicular networks," *China Communications*, vol. 17, no. 5, pp. 266–283, 2020.

[4] W. Wu, F. Zhou, R. Q. Hu, and B. Wang, "Energy-efficient resource allocation for secure NOMA-enabled mobile edge computing networks," *IEEE Transactions on Communications*, vol. 68, no. 1, pp. 493–505, 2020.

[5] X. Sun, J. Zhao, X. Ma, and Q. Li, "Enhancing the user experience in vehicular edge computing networks: an adaptive resource allocation approach," *IEEE Access*, vol. 7, no. 7, pp. 161074–161087, 2019.

[6] X. Zhang, X. Zhu, M. Chikuvanyanga, and M. Chen, "Resource sharing of mobile edge computing networks based on auction game and blockchain[J]," *EURASIP Journal on Applied Signal Processing*, vol. 2, no. 1, pp. 1–23, 2021.

[7] K. Zhang, Y. Mao, S. Leng, S. Maharjan, A. Vinel, and Y. Zhang, "Contract-theoretic approach for delay constrained offloading in vehicular edge computing networks," *Mobile Networks and Applications*, vol. 24, no. 3, pp. 1003–1014, 2019.

[8] G. Li and X. Song, "Data distribution optimization strategy in wireless sensor networks with edge computing," *IEEE Access*, vol. 8, no. 5, pp. 214332–214345, 2020.

[9] C. Yang, Y. Liu, X. Chen, W. Zhong, and S. Xie, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, no. 8, pp. 26652–26664, 2019.

[10] S. Min, W. Yanting, W. Xijun, and L. Jiandong, "Energy-efficient multiuser partial computation offloading with collaboration of terminals, radio access network, and edge server [J]," *IEEE Transactions on Communications*, vol. 68, no. 3, pp. 1524–1537, 2019.

[11] C. Li, W. Chen, J. Tang, and Y. Luo, "Radio and computing resource allocation with energy harvesting devices in mobile edge computing environment," *Computer Communications*, vol. 145, no. 9, pp. 193–202, 2019.

[12] Y. Liao, L. Shou, Q. Yu, Q. Ai, and Q. Liu, "Joint offloading decision and resource allocation for mobile edge computing enabled networks," *Computer Communications*, vol. 154, no. 8, pp. 361–369, 2020.

[13] X. Liu, J. Yu, Z. Feng, and Y. Gao, "Multi-agent reinforcement learning for resource allocation in IoT networks with edge computing," *China Communications*, vol. 17, no. 9, pp. 220–236, 2020.

[14] 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 Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1689–1703, 2020.

[15] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in IoT networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020.

[16] X. Chen, Z. Liu, Y. Chen, and Z. Li, "Mobile edge computing based task offloading and resource allocation in 5G ultra-dense networks," *IEEE Access*, vol. 7, pp. 184172–184182, 2019.

[17] Q. Zhang, L. Gui, F. Hou, J. Chen, S. Zhu, and F. Tian, "Dynamic task offloading and resource allocation for mobile-edge computing in dense cloud RAN," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3282–3299, 2020.

[18] M. Zeng and V. Fodor, "Energy minimization for delay constrained mobile edge computing with orthogonal and non-orthogonal multiple access," *Ad Hoc Networks*, vol. 98, pp. 102060–102060.13, 2020.

[19] Q. Liu, R. Mo, and X. Xu, "Multi-objective resource allocation in mobile edge computing using PAES for Internet of Things," *Wireless Networks*, vol. 5, no. 3, pp. 1–13, 2020.

[20] T. Li, K. Wang, K. Xu, K. Yang, C. S. Magurawalage, and H. Wang, "Communication and computation cooperation in cloud radio access network with mobile edge computing," *CCF Transactions on Networking*, vol. 2, no. 1, pp. 43–56, 2019.

[21] Z. Yang, C. Pan, J. Hou, and M. Shikh-Bahaei, "Efficient resource allocation for mobile-edge computing networks with NOMA: completion time and energy minimization," *IEEE Transactions on Communications*, vol. 67, no. 11, pp. 7771–7784, 2019.

[22] H. Wang, H. Ke, G. Liu, and W. Sun, "Computation migration and resource allocation in heterogeneous vehicular networks: a deep reinforcement learning approach," *IEEE Access*, vol. 8, no. 8, pp. 171140–171153, 2020.

[23] J. Du, F. R. Yu, X. Chu, J. Feng, and G. Lu, "Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 2, pp. 1079–1092, 2019.

[24] H. Tang, C. Li, J. Bai, J. Tang, and Y. Luo, "Dynamic resource allocation strategy for latency-critical and computation-intensive applications in cloud–edge environment," *Computer Communications*, vol. 134, no. 6, pp. 70–82, 2019.

[25] Q. V. Pham, L. B. Le, S. H. Chung, and W. J. Hwang, "Mobile edge computing with wireless backhaul: joint task offloading and resource allocation," *IEEE Access*, vol. 7, no. 99, pp. 16444–16459, 2019.

[26] H. Guo, J. Zhang, J. Liu, and H. Zhang, "Energy-aware computation offloading and transmit power allocation in ultradense IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4317–4329, 2019.

[27] M. Yan, C. A. Chan, W. Li, L. Lei, A. F. Gygax, and C. L. I, "Assessing the energy consumption of proactive mobile edge caching in wireless networks," *IEEE Access*, vol. 7, no. 99, pp. 104394–104404, 2019.