Hindawi
Journal of Robotics
Volume 2022, Article ID 8514235, 9 pages
https://doi.org/10.1155/2022/8514235

Research Article

Network Resource Allocation Strategy Based on UAV Cooperative Edge Computing

Shuo Wang and Ning Kong

School of Energy and Intelligence Engineering, Henan University of Animal Husbandry and Economy, Zhengzhou, Henan 450044, China

Correspondence should be addressed to Shuo Wang; 81239@hnuahe.edu.cn

Received 9 February 2022; Revised 9 March 2022; Accepted 10 March 2022; Published 29 March 2022

Academic Editor: Shan Zhong

Aiming at the problem that fixed mobile edge computing (MEC) server is difficult to meet the needs of mobile users and temporary computing services, this study proposes a network resource allocation strategy based on unmanned aerial vehicle (UAV) cooperative edge computing. First, a UAV-aided MEC scene is designed, and a single UAV with an MEC server is used to provide auxiliary computing services for ground multiusers. Then, an optimization model aiming at total system delay is constructed by considering the system communication model and calculation model. Finally, Deep Q-Network is used to solve the optimization problem to obtain the best resource allocation scheme. Based on the experimental platform, the proposed strategy is demonstrated and analyzed. The results show that when the number of user equipment is 40, the total delay is about 33s, which is 35.29%, 31.25%, and 15.38% lower than other comparison strategies and effectively reduces the computing delay of users.

1. Introduction

In recent years, with the development of 5G mobile communication technology and the Internet of Things (IoT), smart mobile equipment have shown an explosive growth [1–3]. In the context of cloud computing, MEC is considered to be a technical key to improving the computing efficiency of mobile edge equipment [4]. In MEC systems, task offloading is the key for mobile equipment to support resource-intensive applications by executing on edge cloud resources [5, 6]. However, computing servers are usually deployed in fixed base stations. In practical scenarios, fixed base stations cannot meet dynamic services such as user mobility, base station damage, and temporary hotspot areas. It is particularly significant to meet the dynamic communication and multiequipment access requirements.

According to the existing research, computing servers can be deployed on an unmanned aerial vehicle (UAV) to meet the communication requirements. UAV themselves have the advantages of low cost and high mobility. In UAV-assisted MEC networks, mobile equipment can offload tasks to UAV with high computing power and flexible connectivity at the network edge [7, 8]. This method utilizes the flexibility of UAV and better channel gain to offload computing tasks for users, which can not only save computing delay and user energy but also reduce the traffic load on fixed cloud servers [9]. Reference [10] proposed a two-stage joint hovering altitude and power control solution for the resource allocation problem in UAV networks considering the inevitable cross-tier interference from space-air-ground heterogeneous networks.

At present, significant progress has been made on the research of computing task offloading. For example, reference [11] proposed a collaborative service deployment and application allocation algorithm to achieve the final edge service policy deployment. The minimum energy consumption was obtained through the minimum resource ratio increasing algorithm, and computing tasks are redistributed in combination with the load balancing algorithm to balance the computing load. However, its overall computational efficiency needs to be improved. Reference [12] proposed a computing framework for coordinating terminals, edge

nodes, and cloud centers based on the pipeline offloading scheme. According to the computing and communication capabilities of the entire network, they reasonably allocated computing-intensive tasks to specific terminals or clouds, effectively improving computing efficiency. Reference [13] proposed a vehicle-assisted computational offloading architecture for UAVs. The proposed framework used vehicle-assisted computing offload for UAV computing tasks and network resource optimization, which has commonality with the research topic. Reference [14] introduced agents into computing task offloading and proposed a UAV-MEC (UMEC) agent-enabled computing task offloading framework to help users, UAV, and edge clouds perform computing task offloading. Reference [15] proposed a computational offloading scheme to minimize the time and energy consumption of computational task costs. The scheme reduced the execution cost by coordinating the allocation of computing resources between mobile equipment and edge servers. However, many iterations were needed to find the optimal solution. Reference [16] designed a multiround iterative auction algorithm based on auction theory. But the overall performance of the algorithm needs to be further optimized. Reference [17] proposed an optimized auction-based incentive mechanism. The mechanism can optimize long-term system welfare by operating in an online fashion. However, it does not have good scalability for resources that cannot be covered by network hardware.

With the continuous development of computer technology, intelligent algorithms such as machine learning are continuously applied to edge computing. As in reference [18], a heuristic-based algorithm with a low time cost is proposed. Compared with the existing centralized resource allocation and decision-making algorithms, this scheme acquired a higher number of successfully offloaded tasks in different scenarios, but the offloading of computing tasks in complex situations cannot achieve collaborative optimization. Reference [19] proposed a blockchain-driven collaborative framework for MEC. Reference [20] proposed a caching mechanism based on Q-learning to reduce the backhaul traffic load and transmission delay from the cloud. Reference [21] proposed a collaborative computing framework based on deep neural networks. The experimental results indicate that the proposed method has batter effectiveness than the traditional methods. The above methods based on machine learning not only consider the efficiency and timeliness of MEC network resource allocation to a certain extent but also do not have good scalability. Therefore, the offloading strategy based on UAV has been further researched and developed. Reference [22] proposed a deep reinforcement learning-based MEC UAV-assisted computing offloading. The total cost minimization was taken as the objective function. Although this method expands the wireless network to a certain extent, it needs to be further optimized for efficient offloading of computing tasks and reasonable allocation of network resources in complex IoT.

Aiming at the problem that it is difficult for fixed MEC servers to meet the computing needs of mobile users and computing efficiency of existing UMEC, a network resource allocation strategy based on UAV collaborative edge computing was proposed. Compared with the traditional network resource optimization allocation strategies, the innovations of this study are as follows:

(1) For the low-latency service guarantee for users, the UAV carrying MEC server is designed to improve the auxiliary edge computing system model for users. Taking the total system delay as the optimization goal, the transmission delay of user calculation is greatly shortened.

(2) Due to the huge amount of data and high real-time requirements of the 5G system, the proposed strategy uses Deep Q-Network (DQN) to solve the optimal resource allocation scheme, which improves the analysis efficiency of the network state.

2. System Model and Optimization Goal

2.1. System Model. The scenario of human-machine-assisted MEC system uplink communication is studied, as shown in Figure 1, that is, UAVs provide auxiliary computing services for multiple ground user equipment. Only the scenario where a single rotor UAV provides services to N user equipment on the ground is considered.

For the convenience of theoretical analysis, a three-dimensional Cartesian coordinate system is considered, where $L_n = (x_n, y_n)$ represents the position of n user equipment on the ground. The drone flies in a circle within a specified range and provides auxiliary computing services to ground user equipment. Among them, UAV is at a fixed height and the initial position of UAV that is denoted as $d[1]$, and the final position is denoted as $d[K]$. Each ground user equipment offloads some of computing tasks to UAV and the rest is locally computed.

It is assumed that the total execution time is denoted T, and it can be equally divided into K slots and denoted $t = \{1, 2, \ldots, K\}$, where $\tau = T/K$, where τ refers to the length of each slot. In the t slot, the position of drone is $d[t] = (x[t], y[t])$. Assuming that the maximum flight speed of UAV is $V_{\max}$, the trajectory constraints of UAV are as follows:

$$\begin{aligned} d[1] &= d[K], \\ \frac{\|d[t+1] - d[t]\|}{\tau} &\leq V_{\max}, \end{aligned} \tag{1}$$

where formula (1) indicates that the flight speed of the UAV cannot exceed its maximum flight speed. Since UAV flight energy consumption is related to UAV flight speed, UAV flight energy consumption expression is as follows:

$$E^{\text{fly}}[t] = \eta \|v[t]\|^2, \tag{2}$$

where $v[t] = [\|d[t+1] - d[t]\|/\tau]$ represents the flight speed of UAV in the t time slot; η represents the proportional factor of energy consumption and flight speed.

2.2. Communication Model. Assuming that the wireless channel between UAV and each user equipment is a line-of-

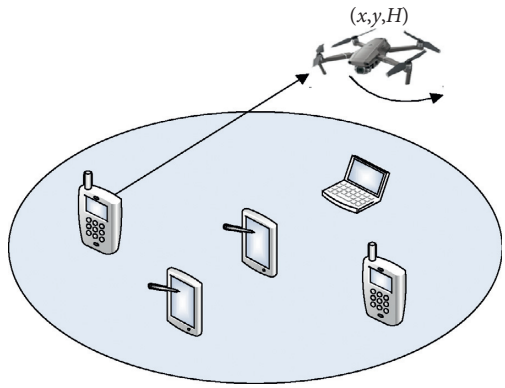

Figure 1: System model.

sight (LOS) channel, the channel power gain between UAV and user equipment n can be expressed as follows:

$$h_n[t] = \beta_0 d_n^{-2}[t] = \frac{\beta_0}{H^2 + \|d[t] - L_n\|^2}, \tag{3}$$

where β_0 represents the channel power gain per unit distance; $d_n^{-2}[t]$ represents the horizontal distance between UAV and user equipment n in the t gap; and $\|\cdot\|$ represents Euclidean norm.

2.3. Computational Model

2.3.1. Local Computing. When user equipment n selects the local mode, all computing tasks will be locally performed, and the computing delay can be expressed by the following formula:

$$T_n^{\text{loc}} = \frac{C_n D_n}{f_n}, \quad n \in N, \tag{4}$$

where f_n is the computing capability of user equipment n (number of cycles per second); C_n represents the number of cycles required to calculate 1 bit; and D_n is the size of input data.

2.3.2. Offload to UAV for Calculation. The UAV carrying MEC server periodically flies at a fixed altitude above the ground user equipment. The user equipment that selects MEC mode can offload tasks to UAV through time-division multiple access (TDMA) [23, 24]. The set of ground users is set to be offloaded to MEC as N', and UAV can choose to relay some tasks to access point (AP) for calculation. Therefore, it can be assumed that the ratio of user equipment n that selects the relay to the ground base station in the t time slot is $\zeta_n[t]$, $n \in N'$, $t \in K$; then, the task that needs to be calculated by UAV in the t time slot is $(1 - \zeta_n[t])D_n[t]$.

(1) The transmission speed of user equipment n in time slot t:

$$v_n^u[t] = B_1 \log_2\left(1 + \frac{P_n h_n[t]}{\sigma^2}\right), \quad n \in N', \tag{5}$$

where B_1 is the bandwidth between UAV and user n; P_n is the maximum transmission power of the user n, and the user equipment uses the maximum power transmission; and $h_n[t]$ is t time slot channel gain between the user n and UAV communication.

(2) Transmission delay of user equipment n: assuming that the bits offloaded by the user n in time slot t are $D_n[t]$, so the user n needs to offload all bits in K time slots, and the constraints are expressed as follows:

$$\sum_{n=1}^{N} D_n[t] = D_n. \tag{6}$$

The total transmission delay of the user equipment is expressed as follows:

$$T_n^{\text{mec}} = \sum_{t=1}^{K} \frac{D_n[t]}{v_n[t]}. \tag{7}$$

(3) The calculation delay of UAV: according to the relay ratio, the task calculated by UAV for user equipment n in the t time slot and then the calculation delay of user equipment in one time slot in MEC mode can be obtained, and finally, the calculation task of UAV for a single user equipment can be obtained. The total delay is as follows:

$$T_k^u = \sum_{t=1}^{K} \frac{(1 - \zeta_n[t])D_n[t]C_n}{f_c}, \tag{8}$$

where f_c represents the computing power of MEC server.

2.3.3. Offload to Base Station on the Ground for Calculation. UAVs can choose to relay part of computing tasks to AP computing. The proportion of UAVs choosing relays in time slot t is $\zeta_n[t] \in [0, 1], n \in N'$. Since the ground base station can have multiple MEC servers built in, the calculation delay of ground base station is considered to be ignored. At the same time, the result of the calculation task is usually very small, and it is considered to ignore the transmission delay of calculation result. Thus, the delay of UAV relaying tasks to the computing part of the ground base station only includes the transmission delay of the UAV.

The flying height of the UAV is relatively high from the ground base station and has a good line-of-sight link. It is considered that the communication between the UAV and ground base station adopts LOS channel. The transmission rate of UAV relaying the task to AP in the time slot is calculated as follows:

$$v_n^{U2E}[t] = B_2 \log_2\left(1 + \frac{P_u h_u[t]}{\sigma^2}\right), \tag{9}$$

where B_2 is the bandwidth between UAV and AP; P_u is the transmission power of UAV; and $h_u[t]$ is the communication channel between UAV and AP.

The transmission delay from UAV relay to ground AP can be expressed as follows:

$$T_n^{U2E} = \sum_{t=1}^{K} \frac{\zeta_n[t] D_n[t]}{v_n^{U2E}[t]}. \tag{10}$$

Assuming that UAV can calculate and transmit part of computing tasks to the ground AP at the same time, the total calculation delay offloaded to UAV is as follows:

$$T_n^{\max} = \max\left\{T_n^{U2E}, T_n^{u}\right\}. \tag{11}$$

When the user equipment selects MEC mode, the total delay of offloading computing tasks to UAV is as follows:

$$T_n^{\Omega} = T_n^{\text{mec}} + T_n^{\max}. \tag{12}$$

2.4. Model Constraint Description. The system goal is to minimize the total delay of all requesting user equipment, so the system goal can be defined as follows:

$$T_{\sum} = \sum_{n=1}^{N} T_n, \tag{13}$$

where T_n represents the calculation delay of the user n.

Considering different computation and offloading modes, T_n is defined as follows:

$$T_n = (1 - \alpha_n) T_n^{\text{loc}} + \alpha_n T_n^{\Omega}, \tag{14}$$

where α_k represents the mode selected by the user equipment. When the variable is 1, it means that the offloading to the drone is selected, and when the variable is 0, it means that the local calculation is selected. T_n^{loc} represents the total delay locally calculated by the user equipment; T_n^{Ω} represents the total delay calculated by the user to select all offload to the drone.

Optimization of user equipment offloading strategy, UAV relay ratio, UAV trajectory, and user bit allocation are combined to minimize the total delay of user equipment. The mathematical expression is as follows:

$$\begin{aligned}
&\text{P1: } \min \sum_{n=1}^{N} \left((1 - \alpha_n) T_n^{\text{loc}} + \alpha_n T_n^{\Omega}\right),\\
&\text{C1: } \alpha_n \in \{0, 1\}, \quad n \in N,\\
&\text{C2: } \zeta_n[t] \in [0, 1], \quad n \in N',\\
&\text{C3: } T_n \le T_n^{\max}, \quad n \in N,\\
&\text{C4: } \sum_{t=1}^{K} D_n[t] = D_n, \quad n \in N',\\
&\text{C5: } d[1] = d[K],\\
&\text{C6: } \|d[t+1] - d[t]\|^2 \le \left(v_{\max}\Delta\right)^2, \quad t \in \tau,\\
&\text{C7: } \sum_{n=1}^{\dot{N}} D_n[t] \le C_u, \quad t \in K,
\end{aligned} \tag{15}$$

where C1 represents that the user equipment n can only choose one mode; C2 represents that the relay ratio of UAV is a variable between 0 and 1. C3 represents that the calculation delay for each user equipment n must be less than the maximum tolerated delay. C4 represents that the user equipment n that selects the MEC mode needs to complete the offloading of tasks within K time slots. C5 means that the drone flies in cycles; C6 means that the maximum horizontal distance of the drone in one time slot cannot exceed the threshold. C7 represents that the bits offloaded by all user equipment to UAV in a time slot cannot exceed the computational threshold of UAV.

3. Solutions Based on Deep Reinforcement Learning

3.1. Reinforcement Learning Modeling. The optimization problem in equation (15) is solved by a single-agent Q-learning algorithm based on reinforcement learning. The algorithm model consists of four parts, namely, agent, state, action, and reward [25].

(1) Agent: Using UAV as the agent of the algorithm, UAV will be responsible for collecting the information of each equipment in the system and making scheduling decisions.

(2) Action: $a = [\lambda, \varpi_n]$ is defined to represent each action decision variable of UAV. Among them, $\lambda = n$ indicates that the drone will provide on-demand services for the n user equipment in the current state, and $\lambda \in N$ is satisfied. ϖ_n represents the computing mode of required computing task of n user equipment.

(3) State: The state variable $s = [\omega, z, t^{\text{serve}}, t^{\text{fly}}]$ is defined to represent the service state in the system. The state quantity consists of the following four parts: the user equipment serves state quantity $\omega = [\omega_1, \ldots, \omega_N]$ and satisfies $\omega_n \in \{0, 1\}$. ω_n represents whether the n user equipment in the system has been serviced by drone and the service status has been completed, $\omega_n = 1$ represents that the drone has completed the on-demand service for the first user equipment, and otherwise, $\omega_n = 0$. The maximum tolerant delay amount $z = [z_1, \ldots, z_N]$ and satisfies $z_n = T_n^{\max}$, that is, z_n represents the maximum tolerant delay of n user equipment in the system. Waiting service delay amount $t^{\text{serve}} = [t_1^{\text{serve}}, \ldots, t_N^{\text{serve}}]$; flight delay amount $t^{\text{fly}} = [t_1^{\text{fly}}, \ldots, t_N^{\text{fly}}]$.

(4) Rewards: The proposed optimization problem aims at maximizing the number of network user equipment services under UAV-MEC system architecture, while the proposed Q-learning algorithm based on reinforcement learning aims to obtain maximum reward feedback. Combining the above two points, and based on the current system state variable, $s = [\omega, z, t^{\text{serve}}, t^{\text{fly}}]$ and the selected action variable $a = [\lambda, \varpi_n]$. The system reward feedback function is defined as $r(s, a) = \sum_{n=1}^{N} \omega_n$, that is, when UAV selects the action decision variable as a in the state s, the total number of network equipment in the system can meet the service requirements [26, 27].

The training iterative process of the Q-learning algorithm based on reinforcement learning satisfies the Bellman equation. The selection principle of action decision performed by UAV under different state variables is based on ε-greedy mechanism.

3.2. DQN-Based Offloading Strategy Optimization Algorithm. The pseudocode of the DQN-based offloading strategy optimization algorithm is shown in Algorithm 1.

First, the network parameters are initialized, including the capacity C_1 of replay experience pool, the parameter θ of action value function $Q(s, a; \theta)$, the parameter $\widehat{\theta} = \theta$ of target action value function $\widehat{Q}(s, a; \widehat{\theta})$, and the initial state s. Then, at each episode t, each user equipment randomly picks an action from the feasible action space according to ε-greedy policy. A random action with probability ε or an action satisfying the formula $a = \arg\max Q(s, a; \theta)$.

Then, resource allocation is performed according to the action selected by user equipment. If the local processing task is selected, let the local CPU cycle frequency be the maximum computing power of equipment, namely, $F_n = f_n^{\text{loc}}$. If the user equipment chooses to upload tasks to UAV for processing, let its uplink transmission power be the maximum available power, i.e., $p_n = P_n$.

After the resource allocation is completed, reward r can be calculated according to the designed reward function, a new state s' can be obtained, and a new sample (s, a, r, s') can be stored in the experience pool. Finally, a batch of samples is randomly sampled from the experience pool, and these samples are used to update the parameters θ of Q-Network, synchronizing the parameters $\widehat{\theta} = \theta$ of target Q-Network every e steps. Note that although each user equipment independently selects actions, since the resources of UAVs are shared, the results of their action selections influence each other [28, 29]. Assuming that all users have selected the same drone for association, the drone may be overloaded, resulting in the user's equipment mission not being able to complete within the specified latency limit, or the drone running out of energy. In order to balance the load as much as possible to avoid this extreme situation, an invalid action pool is added. After all user equipment complete the action selection, if the action set composed of all the user equipment cannot meet the energy consumption constraints of the base station or the load is too concentrated so that most of the user equipment cannot meet the delay limit $T_n^{\max}$, the action set is added to invalid action pool [30]. In this way, when the user equipment performs action selection again, if the current action combination has already appeared in an invalid action pool, it is ignored and the selection is made again. This avoids invalid calculations to improve efficiency.

4. Experiments and Analysis

Simulation results are used to evaluate the performance and efficiency of the proposed strategy. It is assumed that the user equipment is randomly distributed in a two-dimensional area of 200 m × 200 m, and the AP is located in the upper right corner of the two-dimensional area. Each user equipment has different computing tasks, time delay tolerance, and the number of cycles required for computing a 1 bit task. The specific parameters are shown in Table 1.

4.1. Average Reward of Algorithm Training Process. The average reward value of multiple episodes in the training phase of the DQN-based offloading strategy optimization algorithm is shown in Figure 2. In order to increase the exploratory nature, a certain amount of noise is added to the action during the experiment, which makes the original image have many glitches and become unsmooth. Figure 2 shows the reward curve after taking the moving average.

It can be seen from Figure 2 that the proposed algorithm has converged at 2,000 time slots, and the average reward value fluctuates around 220. It can be seen that the algorithm has a fast convergence speed and an ideal average reward value.

4.2. Relationship between the Number of Convergence Iterations and the Number of IoT Equipment. As the number of equipment in the system increases, the number of times that the proposed algorithm training reaches convergence also increases, as shown in Figure 3.

As can be seen from Figure 3, when the number of system equipment increases, more iterations are required to achieve convergence. But when the number of equipment exceeds 35, the increase in the number of iterations increases. This is mainly because the Q-table size of the algorithm is closely related to the number of equipment present in the system. When the number of equipment exceeds the threshold, the processing pressure of the algorithm sharply increases, resulting in a rapid increase in the number of iterations required for convergence. However, the algorithm can achieve convergence regardless of the number of equipment, thus demonstrating the effectiveness of the proposed algorithm.

4.3. Relationship between the Calculated Energy Efficiency and Maximum Energy Consumption of Different Loading Models. The relationship between the computing energy efficiency of user equipment and the maximum energy consumption under different loading models is shown in Figure 4.

As can be seen from Figure 4, the local computing model is that the user equipment only performs local computing, while the global offloading model is that the user equipment completely offloads computing tasks to UAV for computing. Both schemes optimize the flight trajectory of UAV. Using the DQN-based offloading optimization algorithm for partial offloading can achieve higher computational energy efficiency. When the maximum energy consumption is 4J, the computing energy efficiency of its user equipment exceeds 200 bits/J. This is because the user equipment can flexibly allocate resources according to the quality of channel state information, so as to select offload computation or local computation under the partial offload model. Furthermore, the global offload model outperforms the local computing

Initialization Experience pool capacity C_1, Invalid action pool capacity C_2, Parameters θ of action value function Q, Parameters $\widehat{\theta} = \theta$ of target action value function $\widehat{Q}$, Initial state s.
Begin
(1) While $t \leq t_{\max}$
(2) For $n = 1: N$
If rand $(0, 1) < \varepsilon$
User n randomly selects action a from action space A;
Else
User n selection action $a = \arg\max_{a \in A} Q(s, a; \theta)$
End if
End for
(3) Allocate resources according to user actions
(4) For $n = 1: N$
(5) Obtain a reward r and a new status s', and store the new sample (s, a, r, s') in the experience pool.
(6) Update status $s = s'$
(7) Random sampling (s_i, a_i, r_i, s_{i+1}) is conducted from the experience pool to form small batch samples.
(8) Perform gradient descent on $(y_i - Q(s_i, \alpha_i; \theta)^2$ with respect to parameter θ.
(9) Reset the target action value function $\widehat{Q} = Q$ every e steps.
(10) End for
(11) $t = t + 1$
(12) End While
(13) Determine the final user association matrix according to all action sets.
End

Algorithm 1: Pseudocode of offloading strategy optimization algorithm based on DQN.

Table 1: System simulation parameters.

System parameters	Value
Bandwidth between UAV and user/MHz	10
Bandwidth between UAV and AP/GHz	1
UAV computing power/MHz	1200
User task size/Mbits	[20, 30, 50]
User transmission power/W	3
Maximum user delay/s	[10, 15, 25]
Flight cycle time slot of UAV	32
Fixed altitude of UAV/m	60
Maximum available speed of UAV/m·s^{-1}	45
Gauss white noise/W	10^{-10}

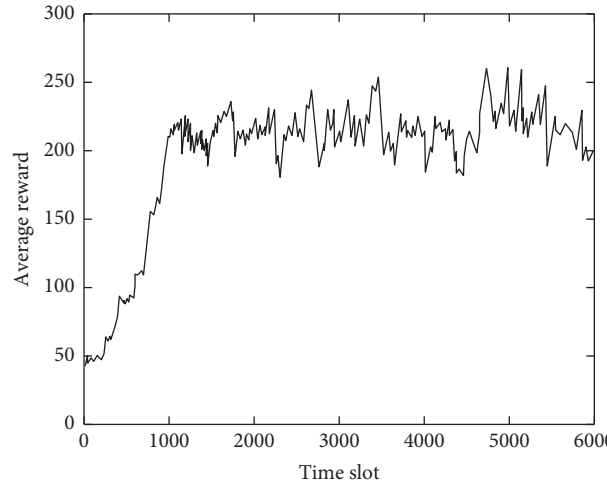

Figure 2: The average reward in the training process of the proposed algorithm.

model, and the computing energy efficiency of user equipment increases with the increase in maximum consumed energy. The reason is that as the energy of user equipment increases, the user equipment has more energy to perform local computations or offload computations. In addition, in the local computing mode, the computing energy efficiency does not change as the energy increases. It can be considered that when the maximum energy consumption value is 1J, the computing energy efficiency of user equipment has reached the maximum value. Subsequently, the user equipment does not need to consume more energy to improve the computing energy efficiency.

4.4. Relationship between Total Delay and the Number of IoT Equipment. As the number of equipment in the system changes, the total latency of the IoT system will also accordingly change. In order to demonstrate the performance of the proposed strategy, it is compared with reference [13], reference [14], and reference [22], and the results are shown in Figure 5.

As can be seen from Figure 5, as the number of equipment increases, the total system delay also increases as expected. The main reason is that when the number of equipment in the system increases, UAV flight delay, UAV edge computing delay, equipment upload delay, and equipment local computing delay corresponding to the newly added equipment will be added to the system. will be added to the total delay of system. Furthermore, the proposed strategy significantly outperforms other comparative strategies in reducing the total system latency. When the

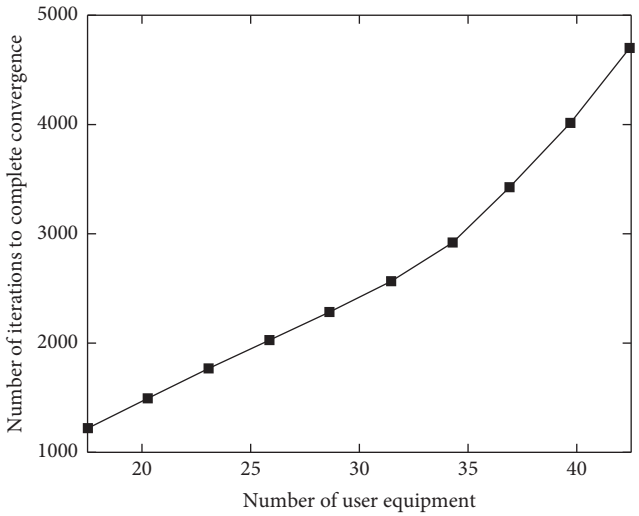

Figure 3: Relationship between the number of convergence iterations and the number of system equipment.

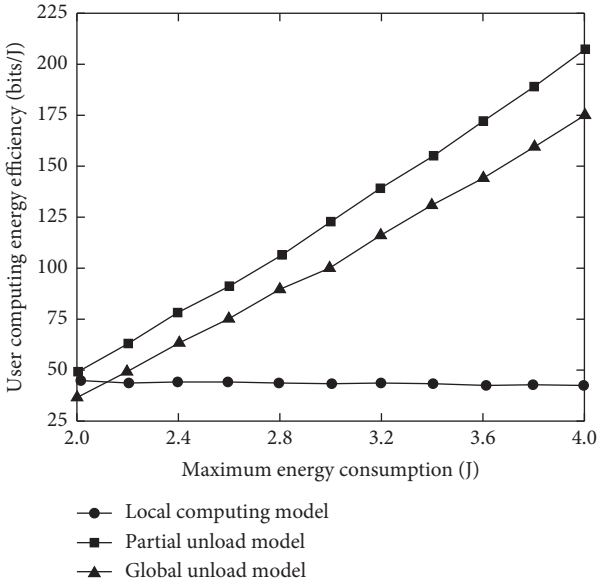

Figure 4: Relationship between calculated energy efficiency and maximum energy consumption of different loading models.

number of user equipment is 40, the total delay is about 33s, which is 35.29%, 31.25%, and 15.38% lower than reference [13], reference [14], and reference [22], respectively. Since the proposed strategy utilizes mobile UAVs for computational offloading and optimizes resource allocation with DQN, the delay can be minimized. Reference [13] proposed a vehicle-assisted computing offloading architecture for UAVs. Since the optimization algorithm has a weak ability to seek optimization, it takes a long time, more than 50s. Reference [14] proposed a UAV-assisted agent-enabled computing task offloading framework to help users, UAVs, and edge clouds perform computing task offloading. However, this algorithm requires a large amount of computation and takes a long time, resulting in increased delay.

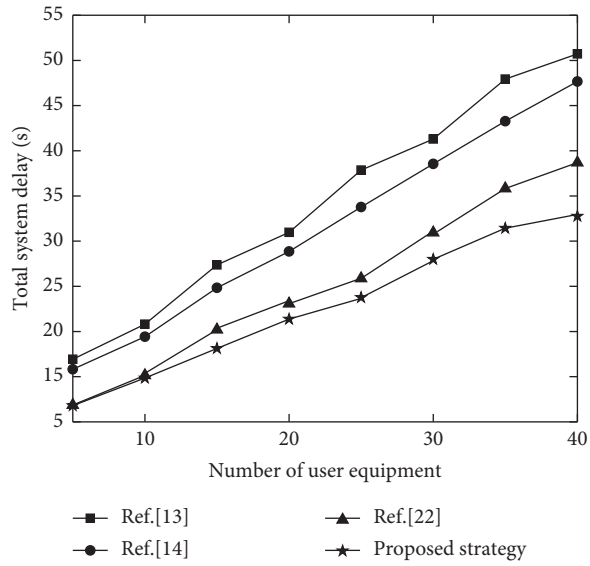

Figure 5: Relationship between time delay and the number of user equipment under different strategies.

Reference [22] proposed a deep reinforcement learning-based MEC UAV-assisted computing offloading scheme. When the number of user equipment is small, the total delay is close to the proposed strategy. However, when the number of user equipment is large, the processing timeliness is not strong, and the delay is relatively large.

5. Conclusions

In order to give full play to the advantages of a UAV-assisted MEC network, it is necessary to further formulate a reasonable user offloading strategy and UAV flight trajectory. To this end, this study proposes a network resource allocation strategy based on UAV collaborative edge computing. Based on the system scenario of collaborative computing between UAVs and ground users, an optimization model is constructed to minimize the total system delay and use DQN to optimize, so as to obtain the best resource allocation scheme. The experimental results show the following:

(1) With the fast mobility of UAVs, users in the system can offload computing tasks in real time, which greatly reduces the transmission delay. Especially when the computing power of UAV reaches 2000 MHz, the total system delay is about 20s.

(2) The proposed strategy uses DQN to optimize the offloading strategy, which not only has a fast optimization speed but also has an efficient optimization result, and realizes a reasonable allocation of resources while reducing the delay. Especially when the amount of user data is large, the optimization effect is more obvious. When the number of user equipment is 40, the total delay is about 33s, which is more than 10% lower than the total delay of other comparison strategies.

As the network scale becomes larger, the network scenarios also become more complex. In order to be suitable for large-scale complex network scenarios and meet the real-time requirements of MEC systems, we will focus on researching and designing a simpler and more effective online scheduling strategy in the future. This is crucial for the large-scale application of MEC.

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.

Acknowledgments

This work was supported by the Provincial Education Reform Project (no. 2020JGLX081) and the Department Level Education Reform Project (no. KCSZ-202022).

References

[1] 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.

[2] M. Sirajuddin, C. Rupa, C. Iwendi, and C. Biamba, "TBSMR: a trust-based secure multipath routing protocol for enhancing the qos of the mobile ad hoc network," *Security and Communication Networks*, vol. 2021, pp. 1–9, 2021.

[3] 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, no. 7, pp. 184172–184182, 2019.

[4] Z. Song, Y. Liu, and X. Sun, "Joint radio and computational resource allocation for noma-based mobile edge computing in heterogeneous networks," *IEEE Communications Letters*, vol. 22, no. 12, pp. 2559–2562, 2018.

[5] W. Fang, S. Ding, Y. Li, W. Zhou, and N. Xiong, "OKRA: optimal task and resource allocation for energy minimization in mobile edge computing systems," *Wireless Networks*, vol. 25, no. 5, pp. 2851–2867, 2019.

[6] Y. Gao, Y. Cui, X. Wang, and Z. Liu, "Optimal resource allocation for scalable mobile edge computing," *IEEE Communications Letters*, vol. 23, no. 7, pp. 1211–1214, 2019.

[7] R. Sridharan and S. Domnic, "Placement strategy for intercommunicating tasks of an elastic request in fog-cloud environment," *Scalable Computing: Practice and Experience*, vol. 20, no. 2, pp. 335–348, 2019.

[8] D. Zhu, C. Xiang, and S. Bing, "Biologically inspired self-organizing map applied to task assignment and path planning of an auv system," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 99, pp. 304–313, 2018.

[9] Z. Fang, J. Wang, and J. Du, "Stochastic optimization aided energy-efficient information collection in internet of underwater things networks," *IEEE Internet of Things Journal*, vol. 9, no. 3, pp. 1775–1789, 2021.

[10] J. Wang, C. Jiang, and Z. Wei, "Joint UAV hovering altitude and power control for space-air-ground iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1741–1753, 2018.

[11] B. Yu, X. Zhang, and I. You, "Collaborative cache allocation and transmission scheduling for multi-user in edge computing," *IEEE Access*, vol. 8, no. 4, pp. 163953–163961, 2020.

[12] C. Kai, H. Zhou, Y. Yi, and W. Huang, "Collaborative cloud-edge-end task offloading in mobile-edge computing networks with limited communication capability," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 2, pp. 624–634, 2021.

[13] M. Dai, Z. Su, Q. Xu, and N. Zhang, "Vehicle assisted computing offloading for unmanned aerial vehicles in smart city," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1932–1944, 2021.

[14] W. A. Rui, C. A. Yong, and B. An, "Agent-enabled task offloading in UAV-aided mobile edge computing," *Computer Communications*, vol. 149, no. 7, pp. 324–331, 2020.

[15] Z. Qin, X. Qiu, J. Ye, and L. Wang, "User-edge collaborative resource allocation and offloading strategy in edge computing," *Wireless Communications and Mobile Computing*, vol. 2020, no. 11, pp. 1–12, 2020.

[16] B. Liu and D. Xu, "Auction-based resource allocation for mobile edge computing networks," *IEICE - Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. 103, pp. 718–722, 2020.

[17] J. He, D. Zhang, Y. Zhou, and Y. Zhang, "A truthful online mechanism for collaborative computation offloading in mobile edge computing," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4832–4841, 2020.

[18] 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. 2, pp. 361–369, 2020.

[19] B. Wu, K. Xu, Q. Li, S. Ren, Z. Liu, and Z. Zhang, "Toward blockchain-powered trusted collaborative services for edge-centric networks," *IEEE Network*, vol. 34, no. 2, pp. 30–36, 2020.

[20] W.-C. Chien, H.-Y. Weng, and C.-F. Lai, "Q-learning based collaborative cache allocation in mobile edge computing," *Future Generation Computer Systems*, vol. 102, no. 6, pp. 603–610, 2020.

[21] E. Li, L. Zeng, Z. Zhou, and X. Chen, "Edge ai: on-demand accelerating deep neural network inference via edge computing," *IEEE Transactions on Wireless Communications*, vol. 19, no. 1, pp. 447–457, 2020.

[22] H. Wang, H. Ke, and W. Sun, "Unmanned-aerial-vehicle-assisted computation offloading for mobile edge computing based on deep reinforcement learning," *IEEE Access*, vol. 8, no. 7, pp. 180784–180798, 2020.

[23] C. Gza, Z. Hao, and B. Yla, "5G network-oriented hierarchical distributed cloud computing system resource optimization scheduling and allocation," *Computer Communications*, vol. 164, no. 4, pp. 88–99, 2020.

[24] M. Mukherjee, S. Kumar, C. X. Mavromoustakis et al., "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6050–6058, 2020.

[25] A. U. Rehman, Z. Ahmad, A. I. Jehangiri et al., "Dynamic energy efficient resource allocation strategy for load balancing in fog environment," *IEEE Access*, vol. 8, no. 2, pp. 199829–199839, 2020.

[26] C. Jiang, Y. Li, and J. Su, "Research on new edge computing network architecture and task offloading strategy for Internet of Things," *Wireless Networks*, vol. 6, no. 2, pp. 1–13, 2021.

[27] L. Ruan, Z. Liu, X. Qiu, Z. Wang, S. Guo, and F. Qi, "Resource allocation and distributed uplink offloading mechanism in fog

environment," *Journal of Communications and Networks*, vol. 20, no. 3, pp. 247–256, 2018.

[28] B. Al-Manthari, N. Nasser, and H. Hassanein, "Congestion pricing in wireless cellular networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 358–371, 2011.

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

[30] T.-M. Pham and T.-T.-L. Nguyen, "Optimization of resource management for nfv-enabled iot systems in edge cloud computing," *IEEE Access*, vol. 8, no. 7, pp. 178217–178229, 2020.