Multi-armed Bandit Algorithm for Online Offloading and Scheduling in Edge Computing Environment

Xiaoying Han
University of Science and Technology of China
hanxiaoying@mail.ustc.edu.cn

Xiaohua Xu*
University of Science and Technology of China
xiaohuaxu@ustc.edu.cn

***Abstract*—In this paper, we study the online scheduling offloading problem in edge computing. Previous scheduling work generally assumes that system have global information, or between the server and the user mutual state information known. Considering the privacy security and the conflicts that users may have during wireless communication, we designed a stochastic multi-played multi-armed bandit online learning offloading framework: D-UCB-G. The framework performs offloading and scheduling based on historical experience, which maximizes the service quality of user devices and also solves the problem that users cannot access the state of edge servers. In the case of ensuring that the user devices do not conflict, the user devices are scheduled to use the edge server resources. We prove the convergence of the algorithm experimentally and demonstrate the effectiveness of the algorithm by applying real-world location data to simulations.**

I. Introduction

With the rise of computing-intensive applications and the increase of terminal devices, edge computing emerges as the times require. Edge computing refers to a new computing model that performs computing at the edge of the network [1]. Data that users may request can be pre-cached on edge devices, and some computing tasks can also be performed on edge. Computing on edge devices can improve the quality of service of end devices without generating large network transmission overhead. However, edge device resources are limited, so a reasonable offload scheduling strategy needs to be proposed to improve the quality of service for users as much as possible. Now most users access the network through wireless networks, we consider the protocol interference model [2], that is, the situation where conflicts may occur between users.

In the research of offload scheduling strategy, some works believe that global information is known [3] [4], and some work believes that edge device status or communication status is known [5]. However, in some cases, the user equipment cannot obtain the status of the edge device, so the offloading selection needs to be made through historical experience. We propose to model the offload scheduling problem with the multi-armed bandit problem. At the same time, user protocol conflicts [8] in wireless communications, which will occur when some users are too close, are always overlooked. Considering the limited edge resources, we also introduced the knapsack constraint.

The multi-armed bandit is a classic reinforcement learning problem. In the problem, each machine provides a random reward from a probability distribution specific to that machine, that is not known a-priori. The objective of the gambler is to maximize the sum of rewards earned through a sequence of lever pulls. Algorithms to solve this problem are widely used in recommender systems and resource allocation [6] [7]. In the edge computing environment, edge devices can be regarded as resources to be allocated. And if each user devices is allocated to the cache or computing resources of edge devices, it will gain certain benefits, such as faster response speed or larger storage space. We design a random multi-play multi-armed bandit online learning offloading framework (D-UCB-G) to solve this problem. The contributions of this paper are as follows:

- We considered protocol conflicts between users in wireless communications and modeled user conflict relationships using graphs. At the same time, taking into account the limited edge resources, we set a knapsack constraint.
- The offloading and scheduling problem is modeled as a MAB problem, and an offload scheduling framework (D-UCB-G) based on UCB algorithm is proposed. It solves the offloading problem and scheduling problem when there are conflicts between users and the state of the edge device is inaccessible. It ensures the maximization of the overall reward.
- The convergence of the algorithm is proved by ex-

979-8-3503-7099-7/24/$31.00 ©2024 IEEE

periments.The established edge computing offloading model is simulated. The results show that the given framework can be optimized and its performance is better than that of the baseline methods.

II. System Model and Problem Formulation

A. System network Model

In this work, we consider a general practical model as illustrated in Fig. 1.We consider that there are N user devices in the edge computing system, expressed as $U = \{u_1, u_2, u_3, \ldots, u_n\}$. There are multiple wireless access points (APs) distributed in various locations of the system, and edge servers are deployed on the APs. Suppose there are J APs in the system, which are expressed as $AP = \{ap_1, ap_2, ap_3, \ldots, ap_J\}$. each AP can directly provide services for user device within a certain radius. It is assumed that the tasks generated by the user device all directly reach the closest AP. It is assumed that users in the system will need to use storage or computing resources such as cached data provided by edge servers during the scheduling process.

This system considers the user's choice of edge servers, and the edge server's scheduling of users according to its environment conditions. And the allocation of communication resources and the allocation of edge server computing resources when the user is offloaded. The tasks that need to be performed by the user device that is not selected for offloading are performed locally.

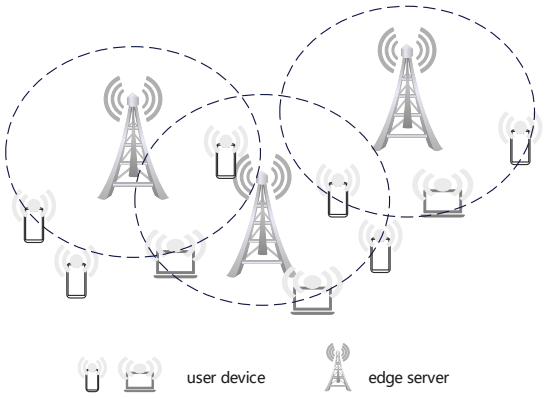


Fig. 1: system model

The system time is divided into consecutive time frames of equal lengths. It is assumed that the total scheduling process has T time periods, and t is used to represent the current time period, $t = 1, ..., T$. Each user device will generate some tasks that need to be executed. In this system, it is set that the user device will only generate one task in each time period. At time t, the task generated by i-th user is $J_i^t \triangleq \{M_i^t, C_i^t, D_i^t\}$, M_i^t indicates the amount of data to be transmitted, in kilobytes. C_i^t indicates the required computing resources. D_i^t indicates the amount of resources required by the task, that is, the abstraction of resources such as CPU and storage. According to knapsack constraint, the sum of the resources of all tasks that need to be offloaded to an edge server must be within a range. We use CAP to represent this range.

In wireless transmission,according to the protocol interference model [2], there may be conflicts between users. Two conflicting users cannot be selected to access the AP at the same time during scheduling. It is assumed that the user does not move during scheduling. We use an undirected graph to represent the conflicting relationship between users. The point set is the user device U. If the edge set E contains (u_i, u_j), it means that the two user devices conflict with each other and cannot be selected at the same time.

B. Wireless communication model

In this system, let the distance between the user device u_n and AP ap_p be $L_{n,p}$.In this model, the code division multiplexing technology is used for the communication path. Referenceing [10] the inter-node signal-to-noise ratio calculation formula and the Shannon's formula, the signal-to-noise ratio between the nth user device u_n and the pth APap_p in the time period t is:

$$SNR_{n,p}(t) = \frac{Power_n \cdot Gain_{n,p}}{\sigma + \sum_{i \in M\{i\}: u_i choose\, the\, AP\, ap_p too} Power_i \cdot Gain_{i,p}}. \tag{1}$$

where $Power_n$ represents the power of the signal transmitted by the user device u_n, $Gain_{n,p}$ represents the communication gain between the user device u_n and AP ap_p, σ is white Gaussian noise, $i \in M\{i\}$: u_i *choose the AP* ap_p *too* indicates that the same AP, that is, the user device of ap_p, is selected at the same time as u_n.we simplify the communication gain to [12]:$Gain_{n,p} = C \cdot L_{n,p}^{-a}$.. where C is a constant factor ,a is the path loss factor.

The transmission speed of task J_n^t can be obtained as:

$$S_n^t = B_w * \log_2 (1 + SNR_{n,p}(t)) \tag{2}$$

where B_w represents the network bandwidth that the AP can allocate.

C. Local execution model

When the task is executed locally, the latency only includes the time of the local execution, and the calculation formula is: $LH_n^t = \frac{C_n^t}{C_{ud,n}^t}$ where C_n^t represents the computation amount of the tasks generated by the end-user device u_n, in the t time period. The amount of computation is the number of CPU cycles required. $C_{ud,n}^t$ represents the CPU cycles executed per second of user device ud.

When the task is executed locally, the energy consumption only includes the energy consumption of the local execution, and the calculation formula is: $LE_n^t = C_n^t \cdot \rho_n$ where ρ_n represents the energy consumed by the user device u_n per unit CPU execution cycle. That is, energy consumption is proportional to the number of CPU execution cycles, and ρ_n is a proportional coefficient.

D. Offload execution model

If the task is offloaded to the AP and edge server for execution. It is generally considered that the amount of result data generated by edge server computing is very small, far less than the amount of uploaded data, so it is ignored. Therefore, the transmission delay only considers the part of the delay for offloading the task to the edge server. The transmission time of task J_n^t to the edge server is: $H1_n^t = \frac{M_n^t}{S_n^t}$. The time required to execute on the edge server is: $H2_n^t = \frac{C_n^t}{C_{es,p,n}^t}$ where $C_{es,p,n}^t$ denotes that the computing resources allocated by the edge server E_p to the task J_n^t can be understood as the computing speed.

Then the total time required to offload the task to the edge server for execution is divided into three parts: 1.Upload the task data to the edge server: $H1_n^t$. 2.The task is executed on the edge server: $H2_n^t$. So if u_n chooses to offload task J_n^t to edge server E_p at time t, the time required is $EH_n^t = H1_n^t + H2_n^t$.

When the task is offloaded to the edge server on the AP for execution, since the task does not need to be executed locally and does not consume local energy, the energy consumed by the edge server is not considered. The energy consumption generated in this case is mainly reflected in the energy consumption in the transmission process. The calculation formula of the energy consumed by the task J_n^t in the transmission process is as follows:$EE_n^t = H1_n^t \cdot p_n$ where p_n is the data transmission power. The transmission time is proportional to the energy consumed by the transmission.

E. Problem formulation

Divide each scheduling process into many time periods. The system in this study is set to have a total of T time periods, and T is known. This study adopts binary offloading method for all tasks. At each time period, offloading and scheduling decisions are made.The offloading policy of the user device u_n in a scheduling system is expressed as follows:

$$OP_n = [o_1, o_2, ..., o_{nn}] \quad o_i = 0 \ or \ 1 \ (i = 1, 2, ..., nn) \tag{3}$$

where nn represents the number of all APs that the user device can access. $o_i = 1$ means that the user device will offload the task to the edge server on the i AP for execution. When $o_i = 1$, $o_j = 0 \ \ (j \neq i)$, because it can only be offloaded to one server. If $o_i = 0 \ \ (i = 1, 2, ..., nn)$ is all 0, it means that the task will be executed locally on the end-user device u_n.

At the same time, if the edge server on AP ap_p is scheduled, then its scheduling policy is expressed as follows:

$$SP_p = [a_1, a_2, ..., a_{pn}] \quad a_i = 0 \ or \ 1 \ (i = 1, 2, ..., pn) \tag{4}$$

where pn represents the number of all end-user devices covered by the edge server service area. All end users are ranked from the nearest to the farthest. $o_i = 1$ means that ap_p schedules the i user to offload tasks to the edge server it deploys for execution.

III. User Scheduling Algorithm Based on UCB

A. MAB problem definition

In this section, the scheduling offload problem is transformed into a MAB problem.The classic MAB problem includes three parts: arm, agent, and reward. Every time the arm is pulled by agent, the arm will return a reward. The MAB question is how to pull these arms to make the long-term reward as high as possible. For the performance of solving the MAB problem, the regret is usually used to evaluate.

1) Arms and Agents: Firstly, the user needs to select an edge server for offloading. So the user u_i itself is the agent, and the server accessible to the user u_i is arms, denoted by $AP^{u_i} = \{ap_1, ap_2, \ldots, ap_{m_i}\}$. In each round, the end-user device select one edge server scheduled to offload, it is equivalent to pulling the arm in the MAB problem.

Then it is the edge server that schedules appropriate end-user devices to offload tasks. So here the user devices $U = \{u_1, u_2, u_3, \ldots, u_n\}$ are used as arms.The end-user

device in the system is fixed and known, with a value of N. Since there are many edge servers in the system, and a single server cannot access all user devices. it is assumed that the end-user devices that can be covered by a certain ap_i during the scheduling process are fixed Invariant, the set is U^{ap_i}, and the number is pn_i. Therefore, the main body of the scheduling is ap_i, the optional arm is the coverable end-user device U^{ap_i} (for brevity, U^{ap_i} will be directly mentioned later), the number of arms is fixed and known as pn_i.Let ap_i choose the number of end-user devices in round t as k_t^i. Because in this scheduling algorithm, there can be multiple end-user devices scheduled in each round, that is, $k_t^i \in [1, pn_i]$. So this is a multiple choice MAB problem.

2) Reward: Then an important part of the MAB problem is the reward. In this scenario, scheduling end-user devices and offloading tasks to edge servers is the process of pulling arm. So the benefit is defined as the benefit of the device offloading. So first of all, the calculation of the benefit considers the improvement of the service quality obtained after the device is uninstalled. It is hoped that the more the service quality is improved, the better. So for the user u_j to offload the task to edge server ap_i, the reward u_j obtained is defined as follows:

$$Reward_i^j(t) = \delta_H(LH_j^t - EH_j^t) + \delta_E(LE_j^t - EE_j^t) \quad (5)$$

where δ_H and δ_E are constant parameters. The first half of the right side of the equal sign is the time saved by executing on the edge server compared to the local execution, which reflects the improvement of service quality from the perspective of delay. The second half is the energy saved by executing on the edge server compared to the local execution, which reflects the improvement of service quality from the perspective of energy consumption.

Abstract the improvement of service quality that users may obtain by using an edge server as the payment for using it. The edge server receives the payment of each user, so the reward calculation is the payment obtained by the use of unit resources usage.the user u_j to offload the task to edge server ap_i, the reward ap_i obtained is defined as follows:

$$Reward_j^i(t) = Reward_i^j(t)/D_j \quad (6)$$

In the MAB problem, the regression is used to reflect the performance of the algorithm. Let each end-user equipment be scheduled to obtain a random variable, and the random variable corresponding to the user equipment u_i is D_i. Let $\mu_i = E[D_i]$, μ represents the expectation of return. Let the set of end-user equipments to be scheduled by ap_i in the t round is $U_{(t,ap_i)}$, and the optimal end-user equipment that can be selected in the t round of scheduling is The set is $U^*_{(t,ap_i)}$. In the MAB problem, the regret is the difference between the income that has been selected and the income obtained by actually choosing the optimal one. According to the proof of the document [11] , the calculation formula is as follows:

$$Regret(t) = \sum_{s=1}^{t} \left[\sum_{j \in \mu^*_{(t,ap_i)}} D_j^t - \sum_{j \in \mu_{(t,ap_i)}} D_j^t \right] \quad (7)$$

The algorithm hopes that $Regret(T)$ is as small as possible. Since in this system, the user's transmission speed and the size of the task will change, the truly optimal choice cannot be determined, and an approximation can only be achieved through learning. It is optimal, so in real situations, the regret cannot be calculated. The goal of the final algorithm is to maximize the overall revenue of the user through the scheduling policy SP_p . The optimization goal is defined as follows:

$$\max_{\pi} \sum_{t=1,\ldots,T}^{T} Reward(t).$$

B. Algorithm

This section introduces a offloading and scheduling framework: D-UCB-G. This framework contains double UCB algorithm. Edge Server implements a multi-played UCB algorithm containing user conflict graphs (MPUCBG). The algorithm is based on the definition of the task offloading problem in the previous section, that is, it is carried out on the basis of abstracting the scheduling problem into a MAB problem.The user conflict graph refers to the $Graph$ established in the system model, and the conflict between end-user devices will be considered when executing the algorithm. Multiple selection is because more than one end-user device may need to be selected each time, so the classic UCB algorithm needs to be improved to make it suitable for the multiple selection in this scenario.

In this algorithm, the calculation of the upper bound of the income follows the calculation method of the upper confidence bound in the classic UCB algorithm. The formula is as follows:

$$\lambda_j = \hat{\mu}_{j,n_j^t} + \sqrt{\frac{2\ln T}{n_j^t}} \quad (8)$$

The first half of the right-hand side of the equation is the expected value of reward. The larger the expected value, the greater the benefit that may be obtained by choosing this arm. The larger the n_j^t in the second half of the right side of the equation, the less the arm is selected. It shows that the exploration of this arm is too little and requires further exploration.This formula is a balance of exploration and exploitation.Therefore, the larger the upper confidence bound, the more worthy of choice.

Then is the calculation of $\hat{\mu}_{j,n_j^t}$. In the classic UCB algorithm, the benefits obtained from all rounds are averaged, but in this system, with the change of the environment, the end user The transmission speed of the equipment, the distribution of tasks, etc. may have changed, so it is believed that the newer the gains are the more real. A scaling factor is added to the original. The formula of $\hat{\mu}_{j,n_j^t}$ is as follows:

$$\hat{\mu}_{j,n_j^t} = (1-g)\hat{\mu}_{j,n_j^t} + gReward(t) \tag{9}$$

where g is the scale factor, indicating the size of the newly obtained reward when it is updated. When $g = 1/n_i^t$, all the gains obtained are averaged with the same weight. When g is a fixed value, the most recent reward is a certain proportion. $Reward_j(t)$ expresses the revenue obtained after the end-user device u_j is offloaded in the t round. Of course, only the devices that are scheduled to offload tasks to the edge server for execution will be updated.

For a certain edge server ap_i, its accessible user equipment is $U_k^{ap_i}$, and the number is pn_k, and its scheduling algorithm flowchart is as follows:

For users to evaluate edge servers applied a UCB-based approach, Reward's calculation refers to 5 ,The algorithm flowchart is as follows:

Algorithm 1 MPUCBG scheduling algorithm

Initialization: initialize the capacity CAP_i. choose each users (collection $U_k^{ap_i}$) once to initialize the $n_i^0 = 1$ and $\hat{\mu}_{i,1}$.
for $t = 1, ..., T$ **do**
  Users selects the edge server according to Algorithm 2; Users who choose U are $U_k^{ap_i}$
  calculate Set:
  $\lambda_i = \hat{\mu}_{i,n_i^t} + \sqrt{\frac{2\ln T}{n_i^t}}$
  sort $U_k^{ap_i}$ by λ_i , save them to a collection $U1^{ap_i}$.
  let $K_t = 1$
  while $U1^{ap_i} \neq \phi$ **do**
    find the $U1^{ap_i}[i]$ with the largest λ to be U_{temp}.
    It's task is J_{temp}
    if $CAP_i - D_{temp} < 0$ **then**
      break;
    end if
    add U_{temp} into the $U_t^{ap_i}$.
    $CAP_i = CAP_i - D_{temp} < 0$;
    Find the users that conflicts with U_temp in conflict graph, delete them in collection $U1^{ap_i}$.delete U_temp;
    K_t++;
  end while
  select the AP(the Edge Server) for $i \in U_t^{K_t}$ by UCB algorithm.
  Schedule all users to get reward.$i \in U_t^{K_t}$ offload tasks to AP, and others execute tasks locally.
  observe $Reward_j(t) \quad j \in U_t^{K_t}$
  update $\hat{\mu}_{j,n_j^t} \quad j \in U_t^{K_t}$.
end for

Algorithm 2 UCB-based Selecting edge server algorithm

Initialization: choose each available AP(edge server,collection AP_i) once to initialize the $n_{e,j}^t$ and $\hat{\mu}_{e,j,n_{e,j}^t}$.
for $t = 1, ..., T$ **do**
  calculate Set:
  $\lambda_j = \hat{\mu}_{e,j,n_{e,j}^t} + \sqrt{\frac{2\ln T}{n_{e,j}^t}}$
  select the ap_j with the largest λ as ap_{temp}
  u_i offload the task to the ap_{temp}
  caculate the reward
  update the $n_{e,j}^t$ and $\hat{\mu}_{e,j,n_{e,j}^t}$ of ap_{temp}
end for

IV. Experiment

A. Convergence

The first is the convergence of the algorithm for the MAB problem. Since in this study, a multiple MAB algorithm is required, the convergence needs to be verified. Several studies have mathematically demonstrated the convergence of the multiple UCB algorithm [11]. But there are still conflicts between users in this study, so it is necessary to verify the convergence of the algorithm in the case of multiple selection and arm conflict. The following figure2 is the situation of each MAB algorithm running when the reward is independent and identically distributed, and the variance is 1 and the mean is 0.The

convergence result is shown in the following The algorithm is convergent and optimal.

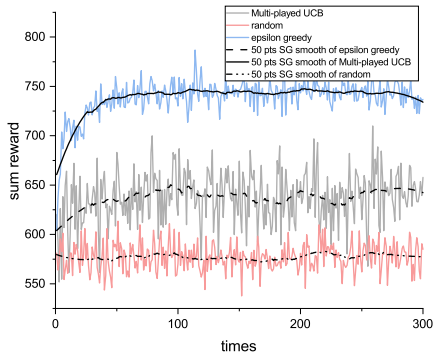


Fig. 2: the convergence of two algorithms for MAB problems: UCB and $\epsilon - greedy$, and the comparison with random selection.

B. Experimental setup

Using a dataset of user devices and edge server locations located in the city of Melbourne [9]. It is composed of 126 base stations and 816 user equipments. The location of the base station is considered to be the location of the edge server. According to the protocol interference model [2] , we think two users conflict if the distance between them is less than 25 meters. Each user will generate a task in each round. The settings of each parameter during the experiment are shown in the tableI.

TABLE I: Parameters

B_w; σ; a; $Power_n$	5Mbps; -100dbm; 4; 200mW
M_i^t; C_i^t;	random [500,1600]KB; [500,1600]Mc;
$C_{ud,n}^t$; D_i^t	random {1,2,3}GHz; [0.3,0.6]
CAP_j; $C_{es,p,n}^t$; p_n	2; 20GHz; 0.5

The baselines set in this experiment are:

- Local: In this case all end-user perform their tasks locally.
- Random1: The D-UCB-G framework is used but the selection is randomly instead of using $lamda$
- Random2: Each end-user randomly whether to uninstall or not. End-users and edge servers choose randomly.

C. Performance analysis

Considering both latency and energy consumption (in (5), δ_H taking 10 and δ_E taking 1), the sum reward of users in each round is shown on (3).It shows that the algorithm is optimized for system offloading and scheduling and will eventually converge.

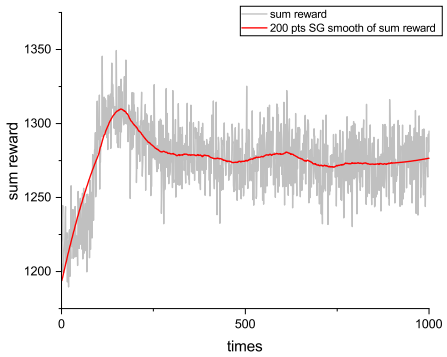
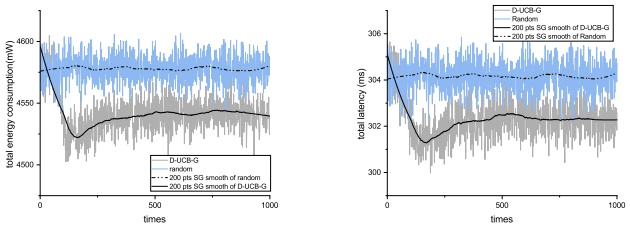


Fig. 3: D-UCB-G conducts 10 experiments, each performing 1000 rounds, and the average per experiment total reward curve of each round of users

Then, the overall latency and energy consumption under different algorithms are investigated. In the case of 3 reward, 10 experiments were performed with 1000 rounds each, and the total latency and energy consumption of users per round using D-UCB-G framework and the Random1 algorithm were shown in 4.It can be seen that The overall latency and energy consumption of the user running task are optimized, and the convergence is achieved. The average delay and energy consumption of the last 200 rounds are shown in the table II. The comparison of Random1, Random2 and Local results shows that using edge server devices in this system setup performs better than only computing locally. The performance of the D-UCB-G framework is better than that of the baseline algorithms, indicating that the service quality to users is improved.



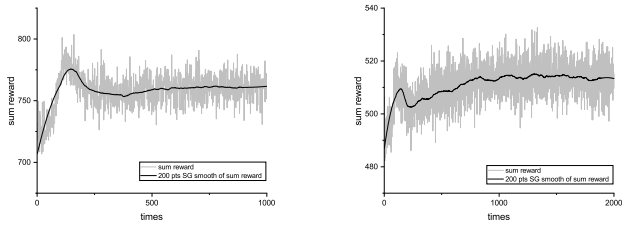
(a) latency (b) energy

Fig. 4: The total reward of users per round using D-UCB-G framework and Random1 algorithms

TABLE II: The averge latency(ms) and energy consumption(mw) of users using D-UCB-G framework and the two baseline algorithms

	D-UCB-G	Random1	Random2	local
energy consumption	4541.95	4577.13	4710.44	5303.73
latency	302.24	304.16	312.58	353.49

In some cases, of course, the system would like to think only about latency or energy consumption. Therefore, when only latency (in 5, δ_H taking 10 and δ_E taking 0) and only energy consumption (in 5, δ_H taking 0 and δ_E taking 1) are considered, the reward in the operation of the algorithm is tested as shown in the following figure 5. When only latency was included in reward, it converge slowly, so 10 experiments were carried out, with 2000 rounds running each time. The latency and energy of the last 200 rounds after convergence of various algorithms are calculated considering delay, energy consumption and both, as shown in the table.



(a) latency (b) energy

Fig. 5: The average per experiment total reward of users of each round

TABLE III: The averge latency(ms) and energy consumption(mw) of users using D-UCB-G framework

	only latency	only energy
energy consumption	4543.75	4540.26
latency	302.07	304.41

Observe TableII and Table III, it can be seen that considering the latency or energy consumption separately, the performance of the system can also be optimized. The latency and energy consumption in the system are not independent of each other, so both will be optimized at the same time. After optimization, compared with the baseline data in Table 2, the system latency and energy consumption are reduced, indicating that the quality of user services has been improved in a targeted manner.

V. CONCLUSIONS

This paper models the edge computing system, and considers the inaccessible state of the edge server, abstracts the offload scheduling problem into the MAB problem, and focuses on improving the service quality of the system to the end users. In view of this problem, a multi-choice UCB offloading and scheduling framework is proposed:D-UCB-G. Experiments show that the framework can be optimized and converged, while having better performance than the baseline algorithm, and can also balance latency and energy consumption through the setting of reward.

REFERENCES

[1] W. Shi and S. Dustdar, "The Promise of Edge Computing," in Computer, vol. 49, no. 5, pp. 78-81, May 2016.

[2] P. Gupta and P. R. Kumar, "The capacity of wireless networks," in IEEE Transactions on Information Theory, vol. 46, no. 2, pp. 388-404, March 2000.

[3] X. Liu, Z. Qin and Y. Gao, "Resource Allocation for Edge Computing in IoT Networks via Reinforcement Learning," ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1-6.

[4] L. T. Tan and R. Q. Hu, "Mobility-Aware Edge Caching and Computing in Vehicle Networks: A Deep Reinforcement Learning," in IEEE Transactions on Vehicular Technology, vol. 67, no. 11, pp. 10190-10203, Nov. 2018.

[5] H. Yuan, G. Tang, X. Li, D. Guo, L. Luo and X. Luo, "Online Dispatching and Fair Scheduling of Edge Computing Tasks: A Learning-Based Approach," in IEEE Internet of Things Journal, vol. 8, no. 19, pp. 14985-14998, 1 Oct.1, 2021.

[6] Gittins, J. C. (1989), Multi-armed bandit allocation indices, Wiley-Interscience Series in Systems and Optimization., Chichester: John Wiley & Sons, Ltd., ISBN 978-0-471-92059-5

[7] Berry, Donald A.; Fristedt, Bert (1985), Bandit problems: Sequential allocation of experiments, Monographs on Statistics and Applied Probability, London: Chapman Hall, ISBN 978-0-412-24810-8

[8] Gupta P, Kumar P. The capacity of wireless networks[J/OL]. IEEE Transactions on Information Theory, 2000, 46(2):388-404.

[9] Lai P. et al. (2018) Optimal Edge User Allocation in Edge Computing with Variable Sized Vector Bin Packing. In: Pahl C., Vukovic M., Yin J., Yu Q. (eds) Service-Oriented Computing. ICSOC 2018. Lecture Notes in Computer Science, vol 11236. Springer, Cham.

[10] X. Chen, L. Jiao, W. Li and X. Fu, "Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing," in IEEE/ACM Transactions on Networking, vol. 24, no. 5, pp. 2795-2808, October 2016.

[11] A. Lesage-Landry and J. A. Taylor, "The Multi-Armed Bandit With Stochastic Plays," in IEEE Transactions on Automatic Control, vol. 63, no. 7, pp. 2280-2286, July 2018, doi: 10.1109/TAC.2017.2765501.

[12] T. Zheng, J. Wan, J. Zhang, C. Jiang and G. Jia, "A Survey of Computation Offloading in Edge Computing," 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), 2020, pp. 1-6.