Privacy-Aware Task Allocation with Service Differentiation for Mobile Edge Computing: Multi-armed Bandits Approach

Hangfan Li[1,2], Lin Shi[1], Xiaoxiong Zhong[1,2,3(✉)], Yun Ji[2], and Sheng Zhang[2]

[1] Guilin University of Electronic Technology, Guilin 541004, China
[2] Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China
[3] Peng Cheng Laboratory, Shenzhen 518000, People's Republic of China
xixzhong@gmail.com

Abstract. With the development of fifth generation (5G) technology, mobile edge computing (MEC) is becoming an essential architecture which is envisioned as a cloud extension version. MEC system can push the resources from cloud side to edge side, aiming to solve many computation intensive problems. The task offloading policy is vital and has an important influence on MEC system. Meanwhile, privacy leakage may occur during the task offloading period which may degrade MEC system performance. The attention on these issues is lack according to existing works. Inspired by this, we present a privacy-preserving aware Multi-Armed Bandits based task allocation algorithm, Privacy Upper Confidence Bound (pUCB), to find a balance between the privacy preserving and the efficiency of task processing. In addition, we take regret analysis of the proposed algorithm. The extensive simulation results show that pUCB scheme can achieve a higher optimal rate, a lower lock rate and less total time cost comparing with traditional Multi-arm bandits (MAB) based algorithm.

Keywords: Mobile Edge Computing (MEC) · Privacy Preserving · Multi-armed Bandits (MAB)

1 Introduction

With the development of the 5G, communication among individuals is becoming more frequency all over the world. Cloud computing has been unable to meet the requirements of task computing or transmission under certain circumstances. MEC [1] has been presented to address these issues, which can bring storage and computation resources closer to edge devices. Computation task will

H. Li and L. Shi—Co-first authors. This work was supported by The Major Key Project of PCL (Grant No. PCL2021A02), National Natural Science Foundation of China (Grant Nos. 61802221) and the Guangdong Talent Project 2021TQ06X117.

F. Gao et al. (Eds.): ChinaCom 2022, LNICST 500, pp. 84–96, 2023.
https://doi.org/10.1007/978-3-031-34790-0_7

be offloaded to edge device, which can reduce delay. Due to the limited computing resources of edge devices, the servers are unable to provide unlimited computational offload services for all tasks among edge devices. Therefore, designing an effective task offloading mechanism and maximizing system performance are challenging issues.

Multi-armed bandits (MAB) framework is wildly used in decision making area which provide a solution of the above issues. In [1], coexisting users are becoming research targets and decentralized task offloading strategies DEBO is proposed to achieve a close-to-optimal performance in MEC system. In [2] and [3], online learning policies based on adversary MAB framework are proposed to deal with peer and flows competition. In [4], MAB based SAGE algorithm is proposed to offer a better performance under different quality of service requirements (QoS). In [5], virtual machines or servers in abundance are used to execute edge computing tasks and Multi-Agent MAB based CB-UCB and DB-UCB are proposed to minimize task computing delay. In [6], an MAB based algorithm is proposed to deal with the uncertainty of MEC system considering energy-efficient and delay-sensitive for a dynamic environment MEC system.

Wang et al. [7] and Gong et al. [8] proposed a novel location-privacy aware service migration scheme and privacy aware online task assignment for MEC. They jointly consider migration cost, user-perceived delay, and the risk of location privacy leakage for making service migration decisions, which is formulated as an MDP process. However, they did not consider differentiated service for MEC. How could we design an efficient resource optimization mechanism for differentiated service MEC when considering privacy preserving, and how could we guarantee the optimal task offloading strategy perform better?

To answer these questions, we present a privacy-preserving aware MAB based algorithm for differentiated service MEC, privacy UCB, which can reach a balance between protecting privacy and task processing efficiency. The contributions of this article are shown as follows:

- We propose an MAB based algorithm for differentiated service (difference type of task like video.mp4 or document.txt) MEC, whose goal are maximizing the resource usage in MEC system and considering privacy preserving at the same time. In proposed scheme, we characterize the delay in task processing of different task types from three computation models: local computation, edge computation and cloud computation. To the best of our knowledge, this scheme is the first work combining MAB algorithms and privacy protecting in differentiated service.
- For privacy preserving in task allocation, we define accumulated privacy quantity (APQ) to quantitative privacy, which aims to reduce the risk of personal privacy leakage. And then, we formulate the optimization problem based on traditional algorithms in differentiated service MEC system. In addition, we take regret analysis of the Privacy Upper Confidence Bound.
- We conduct extensive experiments to evaluate the performance of the proposed scheme. In the simulation results, the proposed algorithm can achieve

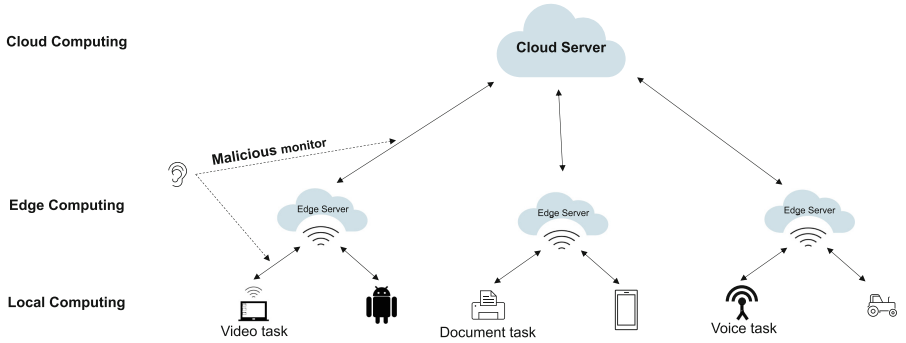

Fig. 1. Traditional MEC System Model

a higher optimal rate, a lower lock rate and total time cost comparing with existing traditional MAB based algorithms.

2 System Model and Problem Formulation

In this section, we firstly introduce our multi-servers computing system based on MEC system. Then we define the delay model for a single task processed by the system. Finally, we propose pUCB for differentiated service in MEC.

2.1 System Model and Useful Notions

In this paper, we consider an MEC system, in where the servers are divided into two categories: N edge servers and one cloud servers indexed by $\{S_1, S_2, \ldots, S_N\}$ where S_1 represents the cloud server and the remaining elements represents the edge servers. We hold a plenty of local devices (LDs) could be denoted by $\{u_1, u_2, \ldots, u_I\}$ and a finite round horizon $\{1, 2, \ldots, t, \ldots, T\}$. The system contains three parts: local devices, edge servers (ES) and one cloud server (CS), which are shown in Fig. 1. Firstly, to simulate reality, LD will generate tasks with random type (task type will be defined later) and size. A task is a command for transmission or handle different types of files or projects. Tasks will be offloaded to a selected server by some strategies where the task can only be computed on edge side, cloud side or local side. Delay-sensitive computing tasks have high requirement on response time, but task size is usually small. Therefore, we assume that a task can be processed before LD moving to another cell MEC system.

2.2 Task Generation

Before considering the privacy quantity, we formulate the overall delay for a task execution process based on the proposed model. For each u_i task will be

generated in each round. Task in each round can be defined as a tuple:

$$h_{i,t}^{\theta} = \left(\mu_{i,t}^{\theta}, r_{i,t}^{\theta}, \tau_{i,t}^{\theta}\right) \tag{1}$$

where $\mu_{i,t}^{\theta}$ represents the size of task, $r_{i,t}^{\theta}$ represents the size of result (usually the returned result size is much smaller than the task size) and $\tau_{i,t}^{\theta}$ represents the time deadline of current task. θ is the type of the task and, for simplicity, in our system, $\theta \in \{$ document, voice, video $\}$.

Once a task has been generated, it should be processed and get feedback. According to the MEC model, we assume that each local device will choose a method to handle the task independently. In this section, we introduce three kinds of computing patterns that can be selected by the local device: local computing pattern, edge computing pattern and cloud computing pattern.

Local Compute Pattern (LCP). Under LCP, the task will be processed by local CPU and get the feedback. So, the delay $d_{i,t,local}^{\theta}$ can be defined as follow:

$$d_{i,t,local}^{\theta} = \frac{\mu_{i,t}^{\theta}}{v_{local}} \tag{2}$$

where v_{local} represents the maximum computing speed of the current LD. For example, $d_{1,2,local}^{voice}$ represents the delay of the task that generated by the first LD in this network in round 2 processing under LCP pattern and the type of task is voice.

Edge Computing Pattern (ECP). Under ECP the tasks will be offloaded to edge servers and the overall process contains three steps: task upload, task execution and result feedback. Task upload delay $d_{i\to n,t}^{\theta}$ and transmission rate $rate_{i\to n}$ can be written as:

$$d_{i\to n,t}^{\theta} = \frac{\mu_{i,t}^{\theta}}{rate_{i\to n}}, n \neq 1 \tag{3}$$

$$rate_{i\to n} = W \log_2 \left(1 + SNR_{i\to n}\right), n \neq 1 \tag{4}$$

where $SNR_{i\to n}$ represents signal to noise ratio and W represents the channel bandwidth. After receiving tasks, edge server starts the task execution process. Task execution delay can be written as:

$$d_{i,n,t}^{\theta} = \frac{\mu_{i,t}^{\theta}}{v_n} \tag{5}$$

where v_n represents the stander capacity of edge server n (bit/sec). Comparing to the cloud server, the capacity of edge server is not strong as cloud server. So the actual capacity of edge server might fluctuate around v_n.

In the end, edge server needs to transmit the result back to the user. So, the download delay from edge server to the local device $d_{n\to i,t}^{\theta}$ and the transmission rate $rate_{n\to i}$ can be written as:

$$d_{n\to i,t}^{\theta} = \frac{r_{i,t}^{\theta}}{rate_{n\to i}}, n \neq 1 \tag{6}$$

So, the task delay under ECP can be defined as follows:

$$d_{ECP}^{\theta} = d_{i\to n,t}^{\theta} + d_{i,n,t}^{\theta} + d_{n\to i,t}^{\theta} \tag{7}$$

Cloud Computing Pattern (CCP). Under CCP, the task will be offload to CS and the overall process contain three steps as same as the ECP: task upload, task execution and result feedback.

The task uploading delay can be written as:

$$d_{i\to 1,t}^{\theta} = \frac{\mu_{i,t}^{\theta}}{rate_{i\to 1}} \tag{8}$$

Due to the long distance between LDs and cloud servers, the signal to noise $SNR_{i\to 1}$ might fluctuate up and down according to the instant network environment.

We assume that the CS is the first server among the server group. After receiving tasks, CS starts the task execution process. Task execution delay can be written as:

$$d_{i,1,t}^{\theta} = \frac{\mu_{i,t}^{\theta}}{rate_{i,1}} \tag{9}$$

where v_1 represents the capacity of the cloud server.

After finishing the tasks, CS needs to give out feedback. So, the download delay can be written as:

$$d_{1\to i,t}^{\theta} = \frac{\mu_{i,t}^{\theta}}{rate_{1\to i}} \tag{10}$$

So, the task delay under CCP, can be defined as follows:

$$d_{CCP}^{\theta} = d_{i\to 1,t}^{\theta} + d_{i,1,t}^{\theta} + d_{1\to i,t}^{\theta} \tag{11}$$

Pattern Selection Vector. A task can only be processed by one computing pattern. So, the pattern selection vector π_t can be written as:

$$\pi_t \in \{\pi_{local}, \pi_{edge}, \pi_{cloud}\} \tag{12}$$

where $\pi_{local} = \begin{bmatrix} 1\,0\,0 \end{bmatrix}$ represents LCP, $\pi_{edge} = \begin{bmatrix} 0\,1\,0 \end{bmatrix}$ represents ECP and $\pi_{cloud} = \begin{bmatrix} 0\,0\,1 \end{bmatrix}$ represents CCP. In addition, the task delay vector can be written as:

$$D_{i,t}^{\theta} = \begin{bmatrix} d_{LCP}^{\theta}\ d_{ECP}^{\theta}\ d_{CCP}^{\theta} \end{bmatrix} \tag{13}$$

therefore, the total delay for a task can be written as:

$$\pi_t \left[D_{i,t}^{\theta} \right]^{\mathrm{T}} \tag{14}$$

where $[\cdot]^{\mathrm{T}}$ represents the vector transpose.

Quantity of Privacy and Problem Formulation. The protection of privacy is an essential problem in today's digital society, and it is also an important index

to measure the quality of algorithms and the overall system. In our system, due to the limited computing resources, it is impossible to deploy the high security encryption algorithm on both user side and server side. Therefore, it is necessary to control the frequency of task uploading to the ES and CS. The risk of privacy leakage is closely bound up with the times of task uploading.

In general, malicious monitors can obtain user (users are LDs in this paper) information from their unloading habits and task unloading probability through illegal ways. Because of the homogeneity of the edge local area network, if the malicious monitor finds a security hole within the system, it is convenient to duplicate the illegal behavior to the whole edge network. Through the combination of these two means, it can be more accurate to identify whether the target user is in the current network, the crime cost is low, and the profit is large.

Therefore, only if we quantify the privacy, we can protect user information. The privacy quantity $q_{i,t}^{\theta}$ can be written as:

$$q_{i,t}^{\theta} = \ln \frac{p_{i,t}^{\theta}}{\bar{p}_{\theta}} \tag{15}$$

where $p_{i,t}^{\theta}$ represents the probability of LD i uploading the task in round t and $\bar{p}_{\theta}$ is the average probability to upload the θ task within MEC system.

If $q_{i,t}^{\theta} > 0$, it means that the task processing method selected by the user has strong personal habit characteristics which is easy to disclose personal privacy. Otherwise, if $q_{i,t}^{\theta} < 0$, it means that current task processing method is universality which helps to reduce the accumulated privacy quantity and reduce the risk of personal privacy leakage. Then the Accumulated Privacy Quantity (APQ) of current LD can be written as:

$$Q_i = \sum_{t=1}^{T} \left(1 - \pi_t \pi_{local}^{\mathrm{T}}\right) q_{i,t}^{\theta} \tag{16}$$

As a result, the final problem for a single LD can be defined as follows:

$$\mathbf{P1}: \text{minimize} \sum_{i=1}^{I} \sum_{t=1}^{T} \pi_t \left[D_{i,t}^{\theta}\right]^{\mathrm{T}}, \forall \theta \tag{17}$$

$$\text{s.t.} \pi_t \in \{\pi_{local}, \pi_{edge}, \pi_{cloud}\} \tag{17a}$$

$$Q_i < Q_{target} \tag{17b}$$

$$D_{i,t}^{\theta} < \tau_{i,t}^{\theta} \tag{17c}$$

$$n \in [1, N] \tag{17d}$$

$$t \in [1, T] \tag{17e}$$

Constraint 1 implies that the task offload approach is limited in three patterns: LCP, ECP, CCP. Based on realistic considerations, constraint 2 and 3 are

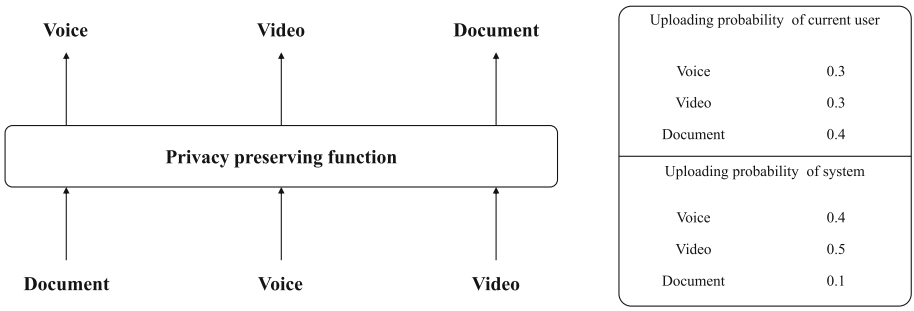

Fig. 2. Traditional MEC System Model

protections on privacy and delay respectively. Constraint 4 and 5 implies that the number of edge server and round horizon are finite. Focused on a normal LD, we need to minimize its total time cost and APQ. Self-locking will occur once the task overtimes to complete and the APQ surpasses its target.

3 Algorithm and Regret Analysis

It is evident that the optimization problem P1 mentioned in section two is easy to solve if we know the key information e.g., server capability, upload-download link state and system resource utilization. However, all this information is not available in advance, and it is infeasible to achieve the optimization in P1 if we result from uncertainty of backhaul information.

We exploit an MAB framework to find a solution to deal with such incomplete priori information situation, which is shown in Algorithm 1. In the MEC system, each user generates different kinds of tasks anytime and anywhere. After each task uploading decisions, the algorithm observe the reward (the shorter time took, the bigger reward got). The size and type of tasks cannot be determined, but the server for processing tasks can be selected, called Action. The collection of actions is called the Action Pool. Hence, the MEC system can learn the distribution from empirical information and determine action selection. We denote $A = \{a_1, \ldots, a_E, a_l, a_c\}$ as the action pool where $a_1 \sim a_E$ represents ECP, a_l represents the LCP and a_c represents the CCP. Inputting such an action pool, we can get the fastest task processing action. Different from classic UCB1 algorithm, in lines 6∼8, it is designed to solve privacy protection problem. According to the definition of quantity of privacy, malicious monitor could only get the regular pattern (often upload some specific type of tasks) of task offloading but not the actual information. Therefore, we can use privacy preserving function to add a misleading task type to protect the regular pattern. As shown in Fig. 2, privacy preserving function could be defined followed confidentiality principle which means the function could totally follow random rule. In our paper, we set the privacy preserving function as follow: set the most common type (the smallest type uploading probability difference between users and the system) as

Algorithm 1. Privacy-UCB algorithm

Input:
An Action pool with K actions;
Try each action $a_m \in A$ in action pool at first;
for $t = K + 1 : T$ **do**
  Estimate the optimistic bound from previous training;
  $a_m = \overline{a_m} + \sqrt{\frac{2 \ln t}{\omega_m}}$;
  $\overline{a_m}$ denote the average delay;
  ω_m denote the number of times of action m has been chosen;
  Try action $a_m = \min a_m$,for all $a_m \in A$
  if $Q_{i,t} \geq \varepsilon Q_{target}$ **then**
    According to the privacy preserving function, add a different type of task as enclosure to mislead the malicious monitors;
  end if
  if $D^{\theta}_{i,t} < \tau^{\theta}_{i,t}$ **then**
    Update $\overline{a_m} = \overline{a_m} + \frac{\left(D^{\theta}_{i,t} - \overline{a_m}\right)}{\omega_m + 1}$;
    Update $\omega_m = \omega_m + 1$;
    Update $Q_i = Q_i + Q_t$
  end if
end for

the misleading enclosure for the most exposed type (the biggest type uploading probability difference between users and the system) of tasks where the probabilities are known. Others are totally random.

Then, we discuss the complexity of the Algorithm 1. Firstly, to initialize the action pool, we need to take an K times iteration, therefore, the complexity of initial part is O(K). For the exploration part, the complexity is related to the action number K which is also O(K). The complexity of main loop, including privacy protection and delay calculation program, is O(N-K). So, the whole complexity of Algorithm 1 is O(N).

Useful Notation and Regret Analysis. Denote $\mathbb{E}[\cdot]$ as expectation and $\mathbb{P}\{\cdot\}$ as probability. Denote K as the number of actions. The time cost of each action obey different distributions and according to the MEC system, we denote $\mathbb{P}_1, \ldots, \mathbb{P}_K$ as the choosing probability for each actions. Denote $G_i(n)$ as the number of times action i has been choose by LD during its first n tasks and μ^* as the shortest time cost under best strategy. Then the regret comparing to the best strategy could defined as:

$$\mu^* n - \mu_j \sum_{j=1}^{K} \mathbb{E}[G_i(n)] \tag{18}$$

Denote $d_{t,s} = \sqrt{2 \ln t / s}$ as the confidence radius distance and $\Delta_i = |\mu_i - \mu^*|$ as the regret of a single round.

Theorem 1. *The average choosing number of $G_i(n)$ of the proposed scheme is limited by an upper bound, which is expressed:*

$$G_i(n) \leq \frac{8\ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3} \tag{19}$$

where n represents the number of tasks.

Proof. We first introduce **Chernoff-Hoeffding Inequality.** [9] Let $\mathbb{X}_1, \ldots, \mathbb{X}_n$ be the random variables with common range $[0,1]$, $\mathbb{E}[\mathbb{X}_t | \mathbb{X}_1, \ldots, \mathbb{X}_{t-1}] = \mu$ and $S_n = \mathbb{X}_1 + \ldots + \mathbb{X}_n$. Then for all $a > 0$:

$$\mathbb{P}\{S_n \geq n\mu + a\} \leq e^{-2a^2/n} \tag{20}$$

$$\mathbb{P}\{S_n \leq n\mu - a\} \leq e^{-2a^2/n} \tag{21}$$

Precisely, for each $t \geq 1$ we bound the indicator function of $I_t = i$ as follows. Let ϕ be an arbitrary positive integer.

$$\begin{aligned}
G_i(n) &= 1 + \sum_{t=K+1}^{n} \{T_t = i\} \\
&\leq \phi + \sum_{t=K+1}^{n} \{T_t = i, T_i(t-1) \geq \phi\} \\
&\leq \phi + \sum_{t=K+1}^{n} \{\overline{\mathbb{X}}^*_{T^*(t-1)} - d_{t-1} \leq \overline{\mathbb{X}}_{i,T_i(t-1)} + d_{t-1}, T_i(t-1) \geq \phi\} \\
&\leq \phi + \sum_{t=K+1}^{n} \{\min_{0<r<t} \overline{\mathbb{X}}^*_r + d_{t-1,r} \leq \max_{\phi<s<t} \overline{\mathbb{X}}_{i,s} + d_{t-1,s}\} \\
&\leq \phi + \sum_{t=1}^{\infty} \sum_{r=1}^{t-1} \sum_{s=\phi}^{t-1} \{\overline{\mathbb{X}}^*_r + d_{t,r} \leq \overline{\mathbb{X}}_{i,s} + d_{t,s}\}
\end{aligned} \tag{22}$$

Then, at least one of the following must hold:

$$\begin{aligned}
\overline{\mathbb{X}}^*_r &\leq \mu^* - d_{t,r} \\
\overline{\mathbb{X}}_{i,s} &\geq \mu_i + d_{t,s} \\
\mu^* &< \mu_i + 2d_{t,s}
\end{aligned} \tag{23}$$

Then according to the Chernoff-Hoeffding Inequality:

$$\mathbb{P}\{\overline{\mathbb{X}}^*_r \leq \mu^* - d_{t,r}\} \leq e^{-4\ln t} = t^{-4} \tag{24}$$

$$\mathbb{P}\{\overline{\mathbb{X}}^*_r \geq \mu^* + d_{t,s}\} \leq e^{-4\ln t} = t^{-4} \tag{25}$$

For $\phi = (8\ln n)/\Delta_i^*$, $\mu^* < \mu_i + 2d_{t,s}$ is false. Therefore, we have

$$r > (8\ln n) \tag{26}$$

$$G_i(n) \leq \frac{8\ln n}{\Delta_i^2} + 1 + \frac{\pi^2}{3} \tag{27}$$

which concludes the proof.

4 Performance Evaluation

In this part, we evaluate the performance of the proposed scheme. The parameters are set as shown in Table 1 and Table 2.

Table 1. Simulation parameters setting for different traffic types.

Type	Offloading probability	Q_i
Voice	0.342	+0.2
Voice	0.342	+0.3
Voice	0.342	–0.05

Table 2. Simulation parameters setting.

Parameter	Value
Local device CPU frequency	2 GHz
Clock cycles required to process a one-bit task	50 clock cycles
Channel bandwidth	1 MHz
SNR between local device and edge server $SNR_{i\to n}$	$2^{20}-1$
SNR between local device and cloud server $SNR_{i\to 1}$	$2^{20}-1$
Task size	1MB $\sim$ 100 MB
Number of edge server	11
Task processing capacity of edge server	10 MB $\sim$ 30MB
Eps-Greedy algorithm	0.4

Optimal rate is the ratio of optimal choosing tasks to overall tasks.

Locking rate is the ratio of device locking number to total number of LD which is set to 100 in our experiments.

The task number is 300, task size is 30MB, the device capacity = 2.0GHz, the epsilon is 0.4, the temperature of SoftMax algorithm is 0.1 and the number of devices is 100. We suppose that there is no privacy concerning limited (QTarget is big enough to ensure device-locking will never happen) if privacy bound is not a variable.

From Fig. 3 (1), the performance of Epsilon-greedy algorithm converges at around 50 tasks which is the fastest one. However, the optimal rate is bad compared with others. SoftMax algorithm converges at around 75 tasks and the optimal rate is better than the previous one. pUCB algorithm and UCB1 algorithm almost have the same performance. Although they converge slowly around

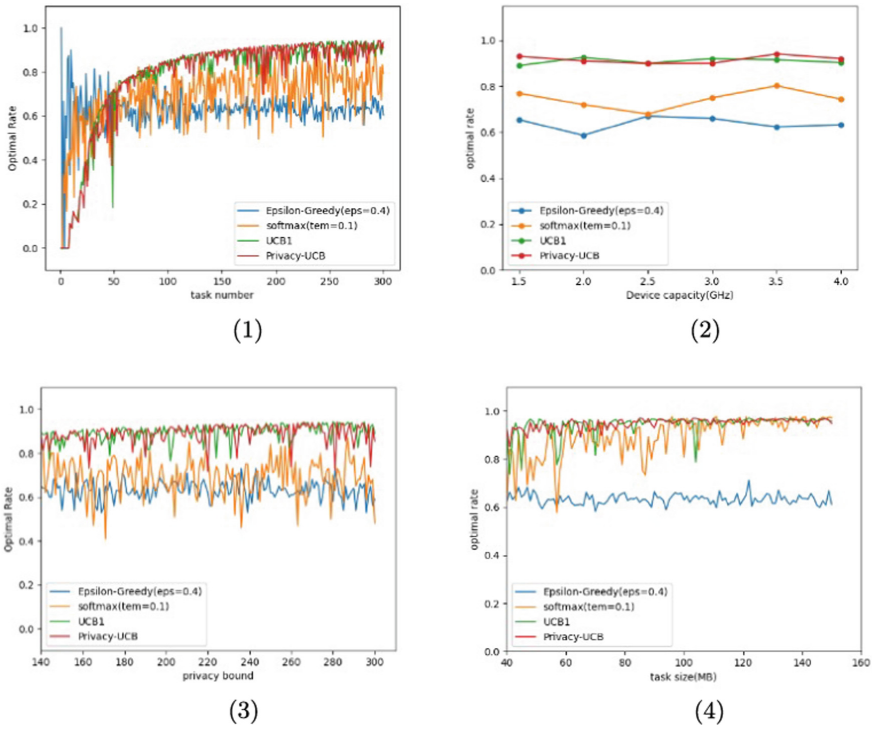

Fig. 3. Optimal rate vs different variables. (1) with task number; (2) with device capacity; (3) with privacy bound; (4) with task size.

100 tasks, but the optimal rate is the best. From Fig. 3 (2), the device capacity has little effect to the optimal rate. In this test, we take the average number of 10 repeated experiments as final report for each CPU frequency. No matter how strong or sick of the CPU performance, local capacity is far less than the server capacity. Therefore, choosing local computing will be a low priority action. The situation of privacy bound and task size is similar with device capacity. From Fig. 3 (3) and Fig. 3 (4), we can confirm the low correlation between optimal rate and these two variables because task size and privacy bound will not influence the capacity of arms directly.

In Fig. 4, it is the most appropriate one to show the difference between pUCB and other traditional MAB algorithms. As we can see, with the increasing of task number, the privacy preserving function plays its due rule. The device lock rate increases rapidly for all algorithms expect pUCB. Combining with the previous test, the desired effect is achieved by using pUCB. The followings are two supplementary trials of the previous one which come to the same conclusion. In Fig. 4 (2) and Fig. 4 (2), device lock rate of pUCB remains zero as the privacy bound increasing and of other three algorithms decline gradually from 100 per-

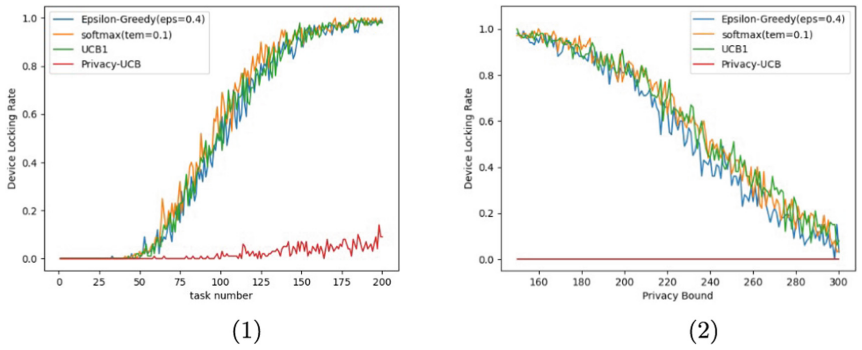

Fig. 4. Device locking rate vs different variables. (1) with task number; (2) with privacy bound.

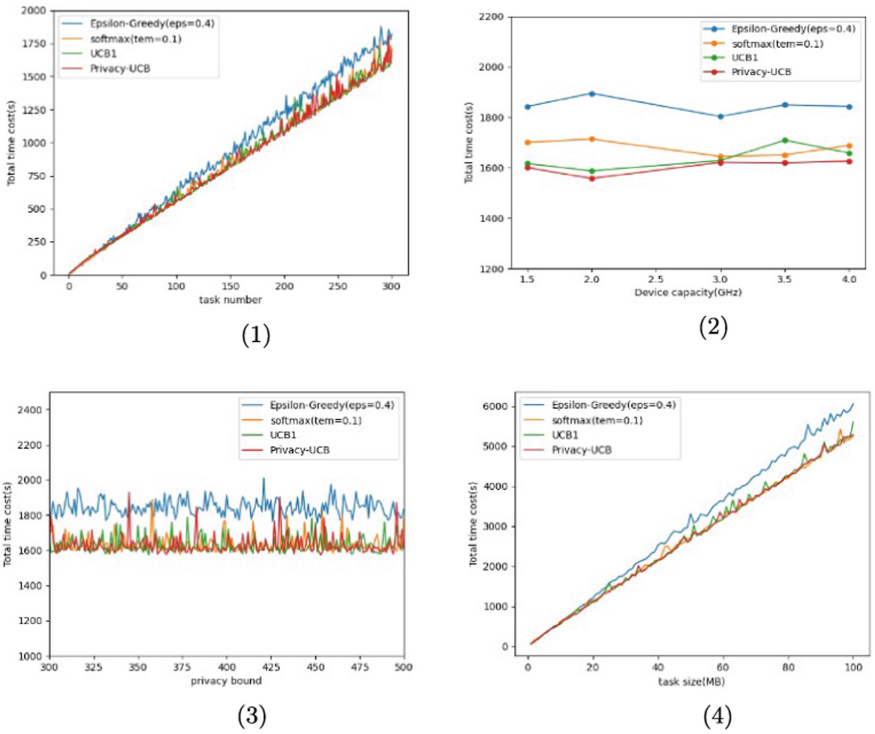

Fig. 5. Total time cost vs different variables. (1) with task number (2) with device capacity; (3) with privacy bound; (4) with task size.

cent to zero. This means, in most cases, pUCB algorithm is a better strategy to acclimatize device to the MEC system.

In Fig. 5 (1), it is a direct confirmation of the previous hypothesis. The time difference required for completion of 300 tasks is 250 s between pUCB and

Epsilon-Greedy (eps = 0.4). We can get the data from the test "total time cost vs device capacity" that the time cost of using Epsilon-Greedy (eps=0.4) algorithm is around 1900 s which is the same with the number in test "total time cost vs task number". In addition, according to the test "total time cost vs privacy bound", if QTarget is big enough to make sure that the device will not lock, the time cost still remains stable and according to the test "total time cost vs task size", it is easy to draw the conclusion that the time cost will grow linearly as the task size growing. From the last four test, we verify pUCB algorithm can get a balance between privacy preserving and algorithm performance in time saving dimension.

5 Conclusion

In this paper, we propose an MAB based algorithm for differentiated service MEC, whose goal is maximizing the resource usage to the MEC system with considering privacy preserving. In proposed scheme, we exploit accumulated privacy quantity (APQ) to characterize task privacy in task allocation algorithm and we prove the regret convergence of the proposed algorithm with an upper bound. We conduct extensive experiments to evaluate the performance of the proposed scheme, and the simulation results show that the proposed algorithm can achieve a higher optimal rate, a lower lock rate and total time cost compared with existing works.

References

1. Zhone, X., Wang, X., Yang, T., et al.: POTAM: a parallel optimal task allocation mechanism for large-scale delay sensitive mobile edge computing. IEEE Trans. Commun. **70**(4), 2499–2517 (2022)
2. Wang, X., Ye, J., John, C.S.: Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach. In: Proceedings of the IEEE Conference on Computer Communications, pp. 1199–1208. IEEE (2022)
3. Hua, C., Wang, L., Gu, P.: Online offloading in dense wireless networks: an adversary multi-armed bandit approach. In: Proceedings of 10th International Conference on Wireless Communications and Signal Processing, pp. 1–6. IEEE (2018)
4. Gao, S., Yang, T., Ni, H., Zhang, G.: Multi-armed bandits scheme for tasks offloading in MEC-enabled maritime communication networks. In: Proceedings of 9th IEEE/CIC International Conference on Communications, pp. 232–237. IEEE (2020)
5. Wu, B., Chen, T., Ni, W., Wang, X.: Multi-agent multi-armed bandit learning for online management of edge-assisted computing **69**(12), 8188–8199 (2021)
6. Ghoorchian, S., Maghsudi, S.: Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty **7**(1), 279–293 (2021)
7. Wang, W., Ge, S., Zhou, X.: Location-privacy-aware service migration in mobile edge computing. In: Proceedings of 2020 IEEE Wireless Communications and Networking Conference, pp. 1–6. IEEE (2020)
8. Gone, W., Zhang, B., Li, C.: Privacy-aware online task assignment framework for mobile crowdsensing. In: Proceedings of 2019 IEEE International Conference on Communications, pp. 1–6. IEEE (2019)
9. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2), 235–256 (2002)