Mobile edge computing scheduling algorithm based on minimum total delay-time

Yuancheng ZHAO1,a

1School of Computer, Guangdong Neusoft Institute, Foshan Guangdong 528225, China
a e-mail: zhaoyuancheng@nuit.edu.cn

Abstract: As an emerging architecture, mobile edge computing (MEC) extends cloud computing services to network edges close to users through MEC servers, satisfying the application needs of real-time control and real-time data analysis. However, due to the limited computing power of MEC server, the task has a long delay-time. In order to improve the current situation, this paper proposes a scheduling algorithm of minimum total delay-time (MTD), in which the server determines the optimal order of task computation to MTD. In addition, this paper proposes an incentive mechanism for users to submit tasks with reasonable computation and expected completion time, while reducing the number and computation of submitted tasks when server computing resources are insufficient. The results show that the performance of the algorithm is close to that of the traditional scheduling algorithm, and total delay-time and average delay-time have improved by 17% to 200%.

1. The introduction

In recent years, with the continuous development of information and communication technology, the number of mobile devices has shown explosive growth, and the mobile data flow has also increased significantly. With the arrival of smart devices and new applications, many new demands have emerged, and the existing mobile cloud computing (MCC) framework has been unable to meet its low latency, high bandwidth, privacy security and other requirements [1-3]. As an emerging architecture, mobile edge computing (MEC) extends cloud computing services to network edges close to users through mobile edge computing servers, and solves problems such as high latency, low bandwidth and privacy security threats in the current cloud computing framework, gradually becoming one of the current research hotspots [4-7].

With the development of artificial intelligence technology, many new technologies that require real-time control and real-time data analysis and are applied to the mobile edge computing framework have emerged, such as smart grid [8], autonomous driving [9] and intelligent medical treatment [10]. However, due to the limited computing power of the edge computing server, the traditional mobile edge computing scheduling method has a long lag time for tasks, which seriously affects the normal operation of the mobile edge computing framework. In addition, the traditional scheduling method does not set an effective incentive mechanism, which makes users submit a large number of tasks or non-essential tasks to the server when the computing resources of the server are insufficient, resulting in a large number of tasks piling up, which seriously affects the real-time performance of computing and unloading services.

Many scholars have studied the unloading scheduling problem of edge calculation.

Ref. [8] proposed a block chain enhanced moving edge computing unloading method based on deep enhanced learning, which considered the block chain data mining task and data processing task,
effectively avoided useless exploration and accelerated the convergence speed without reducing the performance. In Ref. [1], a computational unloading method with data cache in mobile edge computing is proposed, which avoids multiple mobile users from unloading repetitive computing tasks to the network edge and reduces the overall execution delay of mobile terminals. In Ref. [8], a learning-driven unloading method for user asymmetric edge calculation is proposed. By taking advantage of the relevance between task execution time and edge server, lower processing delay is obtained while improving the task unloading rate. However, these works do not take into account the limited computing performance of the mobile edge computing server. If the server encounters a large amount of computation or more tasks, the designed scheduling algorithm will not work properly.

Ref. [5] designed a calculation unloading method based on association and price, and designed a reasonable incentive mechanism based on game theory to ensure the efficiency of calculation unloading. Ref. [7] proposes a software-defined task unloading method using MEC in ultra-dense networks, which minimizes delay and saves battery life of user devices. Ref. [10] designed an energy-aware computing unloading method for the Internet of things in a heterogeneous network, proposed an iterative solution framework, and formulated a transmission power distribution strategy and a calculation diversion scheme. However, these works do not take into account that users in the framework of mobile edge computing still submit a large number of tasks or submit non-necessary tasks to the server when the computing resources of the server are insufficient, resulting in a large number of tasks piling up, which seriously affects the real-time performance of computing unload service.

To solve the above problems, this paper proposes a dynamic programming based unloading scheduling strategy to minimum total delay-time (MTD) in MEC. In this strategy, users submit tasks to the edge computing server at the same time as the expected completion time. The edge computing server schedules tasks according to the amount of data submitted and the expected completion time, and determines the optimal order of task calculation in polynomial time. At the same time, this paper also proposes an incentive mechanism, which enables users to submit tasks with reasonable calculation amount and expected completion time, and reduces the number and calculation amount of submitted tasks when the server computing resources are insufficient. The experimental results show that the performance of the algorithm is close to that of the traditional scheduling algorithm, while the total and average latency have increased by 17% to 200%.

2. Problem description

This paper considers the mobile edge computing framework shown in Fig.1, which has $n$ users and 1 mobile edge computing server. Each user submits a calculation unloading task to the mobile edge server, which feeds back the calculation results to the user through the high bandwidth link after the calculation.

![Framework of MEC](image_url)

Users can use collections $U = \{u_1, \cdots, u_n\}$ Represents its submitted computation offloading task as $T = (P, D) = \{t_1, t_2, \cdots, t_n\}$, of which $P = \{p_1, p_2, \cdots, p_n\}$ Represents the computation time required for the task, $D = \{d_1, d_2, \cdots, d_n\}$ Represents the expected completion time for the task. Mandate the lag time is
$L = \{l_1, l_2, \cdots, l_n \}$ if the mission is $d_0$; Prior to completion, the delay in the task $l_i = 0$; if the mission is $d_i$.

The MEC server is represented by $M$. It is deployed near the network edge of the user. The delay between the server and the user is low enough and the link bandwidth is high enough. The transmission time for the user to submit tasks to the server and the server to return results to the user can be ignored. However, the computing performance of MEC server is limited, and only one computing offload task can run at a time, so it is necessary to determine the optimal order of task computing to MTD.

To sum up, the optimization objective of this paper is to MTD under the condition that the MEC server can only simultaneously calculate a task and the task cannot be interrupted after the start of the task:

$$
\min \sum_{i=1}^{n} l_i \\
\text{s.t. } f_i \leq \sum_{j=0}^{n} p_i, \forall i \\
\quad s_i + p_i = f_i, \forall i
$$

3. Scheduling algorithm

The MTD scheduling algorithm based on dynamic programming needs to determine the computing tasks $\{t_1, t_2, \cdots, t_n\}$ the optimal order to obtain the MTD. Use $k$ Represents the tasks of these tasks with the maximum computation time. So, for one $\delta (1 \leq \delta \leq n - k)$ There is an optimal schedule [17] the optimal schedule can be seen as a series of three subsets of tasks:

1. A sequence of tasks $\{t_1, t_2, \cdots, t_k, t_{k+1}, \cdots, t_{k+\delta}\}$;
2. Task: $t_k$;
3. A sequence of tasks $\{t_{k+\delta+1}, t_{k+\delta+2}, \cdots, t_n\}$;

Among them, task $t_k$ the completion time $f_k(\delta)$ for:

$$
f_k(\delta) = \sum_{j=\delta}^{\delta+k} p_j
$$

Obviously, for the overall scheduling to be optimal, the first and third task subsets need to be internally optimal as well. Therefore, a scheduling algorithm based on dynamic programming is designed to determine the optimal order of task set after determining the final order of task subset of task set. Set up a subset $T(j, n, k)$ represents all tasks $\{t_j, t_{j+1}, \cdots, t_{n-k}\}$ less than equal task $k$ computing time $p_i$ of all missions, including missions $k$ not part of this subset, even if $j \leq k \leq n$ not when. $V(T(j, n, k), \gamma)$ represents the subset in time $\gamma$ the total lag time for that subset in an optimal order of beginning.

To sum up, the MTD scheduling algorithm based on dynamic programming is described as follows. Initial conditions:

$$
V(\emptyset, \gamma) = 0 \\
V(\{j\}, \gamma) = \max(0, \gamma + p_j - d_j)
$$

This initial condition is expressed if the subset is at any time $\gamma$ when empty, the total delay-time is 0. If the subset has only one task, the total delay-time is the delay-time of that task. Recursive relation:

$$
V(T(j, n, k), \gamma) = \min(V(j, k' + \delta, k'), \gamma) + \max(0, f_{k'}(\delta) - d_{\gamma}) + V(T(k' + \delta + 1, n, k'), f_k(\delta))
$$

where, $k'$ is the task with the maximum computing time in the subset, namely:

$$
p_k' = \max(p_j / j' \in T(j, n, k))
$$

The initial equation at the beginning of scheduling is:
The worst-case time complexity of the algorithm can be determined as follows. There are at most \( O(n^3) \) task subsets \( T(j,n,k) \) and \( \sum p_j \) time points in time \( \lambda \), so this algorithm needs to solve at most \( O(n^3\sum p_j) \) recursive equations. The time complexity of solving each recursive equation is \( O(n) \), so the computational complexity of this algorithm is bounded by \( O(n^4) \), which is obviously a polynomial of \( n \). Due to the small scale of users in edge computing, the time complexity of the algorithm is \( O(n^4) \).

### 4. Incentive mechanism

Due to the limited computing capacity of the edge server, if there are many tasks waiting in the server, the tasks submitted later will have a very large delay-time, which will seriously affect the normal operation of the MEC framework. Therefore, it is necessary to set a reasonable incentive mechanism according to the number of tasks currently waiting for the server so that users can minimize the submission of tasks or reduce the data volume of tasks when the server is waiting for tasks.

In addition, tasks submitted by users with a long computing time will occupy more server computing resources, and the expected completion time of submitted tasks is close to the computing time, which will consume the nearest computing resources of the server, which is the most valuable. Therefore, in order to make users submit tasks with less data amount and submit reasonable expected completion time of tasks, a reasonable incentive mechanism should also be set.

Let the current time slice calculate the time required for the task waiting in the server as \( P \), and the value of \( P \) is the common knowledge of all users, and it is temporarily updated at each time slice. The larger the \( P \) value, the greater the cost for users to submit tasks with the same amount of calculation and expected completion time.

To sum up, the server unloads the task to the computation fees charged \( C_i \) for:

\[
C_i = \alpha p_i \cdot \beta^{k-n} \cdot e^{\nu P}
\]  

where \( \alpha \geq 1 \), \( 0 < \beta \leq 1 \), \( \nu > 0 \).

Since the user's costs are closely related to the server's free computing resources, the amount of computation of the task, and the expected completion time of the task, the collection of \( C_i \) can motivate users to submit reasonable computation offloading tasks and ensure the normal operation of the MEC framework.

### 5. Experiment and performance analysis

In this paper, we compare MTD with First-in-first-out scheduling algorithm (FIFO), Random scheduling algorithm (RANDOM) and branch and bound scheduling algorithm (BB) in three aspects: total latency, average latency and computation time of scheduling sequence. Among them, the total latency and the average latency represent the total and average latency of user tasks in edge computing applications respectively. Only with reasonable latency can edge computing applications operate normally. Scheduling sequence computing time represents the performance of the algorithm. Only when the task arrives, the scheduling algorithm can quickly determine the scheduling sequence to ensure the reliability of the algorithm in practical application.

#### 5.1. Experimental Settings

In the experiment, the edge computing server was a MEC server with 8 core 4.0ghz, 32 GB of memory and 1 TB of hard disk. Every time the server performs the scheduling task, each algorithm performs 100 times, and the average value is taken as the final experimental result. The number of users is set to 5~25, in line with the applicable scenario of MEC for small-scale users. Each user submits a computational unload task with a different amount of computation and expected completion time.
5.2. Result analysis
In the simulation experiment, the calculation amount of the task is set as 100–500, and the expected completion time of the task is the calculation time plus 0–100.

![Fig.2 Comparison of total delay time](image1)

![Fig.3 Comparison of average delay time](image2)

![Fig.4 Comparison of scheduling order calculation time](image3)

As shown in fig.2, MTD algorithm has the smallest total lag time, followed by BB algorithm, but the total lag time increases by 17% to 200% compared with MTD algorithm. FIFO algorithm and RANDOM algorithm have the worst performance, and the total lag time increases by 53% to 160% compared with BBM algorithm. In addition, the MTD algorithm has an obvious advantage over the other three algorithms in total lag time when the number of users increases.

As shown in fig.3, the average latency of MTD algorithm is also the smallest, followed by BB algorithm, but the average latency increases by 17% to 200% compared with MTD algorithm. FIFO algorithm and RANDOM algorithm are also the worst performing, and the average latency increases by 14% to 173% compared with BB algorithm. With the same total lag time, the MTD algorithm has an obvious advantage over the other three algorithms in average lag time when the number of users increases.

As shown in Fig.4, MTD algorithm is faster than BBM algorithm, random algorithm and FIFO algorithm in scheduling sequence calculation time. Compared with BB algorithm, MTD algorithm's time increment of scheduling sequence calculation is within 50ms, and compared with FIFO and random algorithm, MTD algorithm's time increment of scheduling sequence calculation is within 100ms. In addition, because the computation complexity of MTD algorithm is \(O(n^4)\), the larger the number of tasks, the more obvious the computation time of scheduling sequence will increase. However, the applicable scenario of mobile edge computing is small-scale users, and the number of users is small.
The scheduling sequential computing time of MTD algorithm will not affect the normal operation of MEC framework.

6. Conclusion
In this paper, the traditional MEC scheduling algorithm is analyzed, and it is found that the existing algorithm does not take into account the limited computing performance of the server, which leads to a long task delay-time in the practical application. Therefore, a scheduling algorithm based on dynamic programming with MTD is proposed. In this algorithm, the user submits the expected completion time of the task at the same time of submitting the calculation unload task, and the server calculates the optimal order of task calculation based on the data volume and expected completion time of the task to MTD. At the same time, this paper also proposes an incentive mechanism to enable users to submit tasks with reasonable computation amount and expected completion time, while reducing the number and computation amount of submitted tasks when server computing resources are insufficient. The experimental results show that, on the premise that the performance of the algorithm is close to that of the traditional scheduling algorithm, the total delay-time and average delay-time of the algorithm have improved by 17% to 200% compared with the traditional scheduling algorithm, which has very important practical significance and can be used in industrial Internet of things, smart home, smart medical and other applications.

Reference
[1] Pan J, Mcelhannon J. Future Edge Cloud and Edge Computing for Internet of Things Applications[J]. IEEE Internet of Things Journal, 2017, 99(5):439-449.
[2] Syed, Noorulhassan, Shirazi, et al. The Extended Cloud: Review and Analysis of Mobile Edge Computing and Fog from a Security and Resilience Perspective[J]. IEEE Journal on Selected Areas in Communications, 2017,35(11):2586-2595.
[3] Du B W, Huang R H, Xie Z P, et al. KID Model-Driven Things-Edge-Cloud Computing Paradigm for Traffic Data as a Service[J]. IEEE Network, 2018, 32(1):34-41.
[4] Wang S, Zhang X, Zhang Y, et al. A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications[J]. IEEE Access, 2017,99(5):6757-6779.
[5] Mach P, Becvar Z. Mobile Edge Computing: A Survey on Architecture and Computation Offloading[J]. IEEE Communications Surveys & Tutorials, 2017,19(3):1628-1656.
[6] Mao Y, You C, Zhang J, et al. A Survey on Mobile Edge Computing: The Communication Perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4):2322-2358.
[7] Abbas N, Zhang Y, Taherkordi A, et al. Mobile Edge Computing: A Survey[J]. IEEE Internet of Things Journal, 2017, 99(5):450-465.
[8] Chen S, Wen H, Wu J, et al. Internet of Things Based Smart Supported by Intelligent Edge Computing [J]. IEEE Access, 2019,6(7):74089-74102.
[9] Feng J, Liu Z, Wu C, et al. AVE: Autonomous Vehicular Edge Computing Framework with ACO-Based Scheduling[J]. IEEE Transactions on Vehicular Technology, 2017,66(12):10660-10675.
[10] Krishnan P R, Durga P, Srisra R E. IoT Based Smart Edge for Global Health: Remote Monitoring with Severity Detection and Alerts Transmission[J]. IEEE Internet of Things Journal, 20186(1):2449-2462.