A Priority Task Offloading Scheme Based on Coherent Beamforming and Successive Interference Cancellation for Edge Computing

Zhehao Li, Lei Shi(✉), Xu Ding, Yuqi Fan, and Juan Xu

School of Computer Science and Information Engineering,
Intelligent Interconnected Systems Laboratory of Anhui Procince,
Hefei University of Technology, Hefei 230009, China
shilei@hfut.edu.cn

Abstract. In edge computing environment, edge servers are generally more closer to edge devices which can guarantee time sensitive tasks be completed under their strict requirements. However, with the rapid increase of edge devices and the limited computing resources of edge servers, this guarantee is becoming more and more difficult. In this paper, by using two physical layer techniques, we try to give communication tasks more opportunities for executing under edge computing environment. In specific words, we propose a priority task scheduling scheme based on coherent beamforming (CB) technique and successive interference cancellation(SIC) technique. CB technique give edge devices the chance to be transmitted to distant edge servers, and SIC technique give communication tasks more chance to be received by edge servers. However, these two techniques need some strict conditions for realizing, and if we consider the computing work and the communicating work simultaneously, the problem will become very complex. We first build the system model and analyze it, and show the model a NP-hard problem and cannot be solved directly. Then in our algorithm, we first determine the task transmission of each time slot in turn, and set the fitness threshold so that each task can select the most suitable edge server. After tasks arrive at servers, we insert them into task queues according to their priorities. In simulations, we compare our scheme with other three schemes. Simulation results show that our scheme can improve the task completion rate and reduce completion delay.

Keywords: Edge computing · Task offloading · Coherent beamforming · Successive interference cancellation

Supported by the National Natural Science Foundation of China (Grant No. 61806067), the Anhui Provincial Key R&D Program of China (202004a05020040).

1 Introduction

Comparing with the cloud computing [1], in the edge computing structure [2,3], servers are positioned at the edge of the network, which can greatly reduce the distances between edge devices and edge servers [4]. In this way, if edge devices need to unload tasks to servers for computing, there is no need for long-distance transmissions [5]. This will reduce the transmission delay [6] and improve the quality of service [7,8]. However, unlike cloud servers, edge servers usually have limited power. Meanwhile many application tasks are time sensitive and must be completed within specified times [9]. Therefore, when the number of tasks is large, a single edge server may not be able to ensure all tasks completed within the specified time. In this case, the server usually migrates tasks to other edge servers for computing [10,11]. But this will increase the task executing cost, especially for the edge computing environment with 5G techniques [12]. Since the 5G base station has a small coverage compared with the 4G's.

The coherent beamforming (CB) technique is one kind of cooperative communication technologies which are usually used on the transmitters. In [13], authors use distributed coherent communication techniques to enable extended-range communications. In [14], authors proposed a CB technique scheme with minimum transmitting power for multi-input-single-output (MISO) communication with limited rate feedback. In [15], for long-distance communication, the author proposes a distributed optimization solution based on CB technique, which significantly improves the network performance. In [16], authors derive an approximation of the coverage probability of the CB scheme by leveraging two scaling factor. This means that edge devices can directly transmit tasks to distant targets and avoid transmission delay. In this way, we can directly transfer tasks to edge servers with sufficient computing resources.

Unlike the CB technique, the successive interference cancellation (SIC) technique is used on the receivers. Since in the edge computing environment, many edge devices may transmit simultaneously and these may cause lots of interference. By using SIC technique, edge server can accept multiple signals at the same time, and decode the signals in turn according to the signal-to-noise ratio(SINR). In [17], authors propose a heuristic algorithm and use SIC to obtain a bandwidth sensitive high throughput protocol. In [18], authors propose a cross-layer optimization framework for SIC that incorporates variables at physical, link, and network layers, and prove the validity of this framework. In [19], authors proposed a neighbor discovery algorithm based on SIC technique.

In this paper, we combine CB and SIC technique, and apply it to the task offloading in edge computing environment. Because there will be multiple tasks in the server at the same time, the newly arrived task is unlikely executed immediately. These tasks will be stored in the server's task queues [20]. Therefore, it is extremely important to select a suitable position for the task in the queue. Many researchers build task queues based on the order in which tasks arrive at the server, and we will choose the appropriate position of new arrival tasks in the queue according to the task priority. The main work of this paper as follow: (1) In a two-dimensional network with multiple nodes and servers, we use CB technique at transmitter and SIC technique at receiver. We select the appropriate processing server for each task. (2) In edge server, we build the queue according to the actual situation of task. Our goal is try to make all the tasks done within the specified time.

The rest of this paper is organized as follows: In Sect. 2, we build the system model. In Sect. 3, we design a scheduling algorithm according to the model and try to get a feasible solution. In Sect. 4, we give the simulation results in different environments and analyze them. In Sect. 5, we summarize the whole paper.

2 System Model

Consider an edge computing network is consisted with m edge servers and u edge devices in a two-dimensional area(see in Fig. 1). Define D as the set of edge devices and $d_i(d_i \in D, 1 \leqslant i \leqslant u)$ as one edge device. Suppose all devices have the same transmission power P. Define S as the set of edge servers and $s_j(s_i \in S, 1 \leqslant j \leqslant m)$ as one edge server. We divide the whole scheduling time T into h time slots, and denote $t_k(1 \leqslant k \leqslant h)$ as one time slot. Suppose edge devices will generate tasks randomly, and suppose all tasks need to be uploaded for calculating. Define V as the set of tasks and $v_{pi}[k](1 \leqslant p \leqslant n)$ as one task, where p means the task is the p-th task generated in the whole network, i and k indicate that the task is generated by edge device d_i in the time slot t_k.

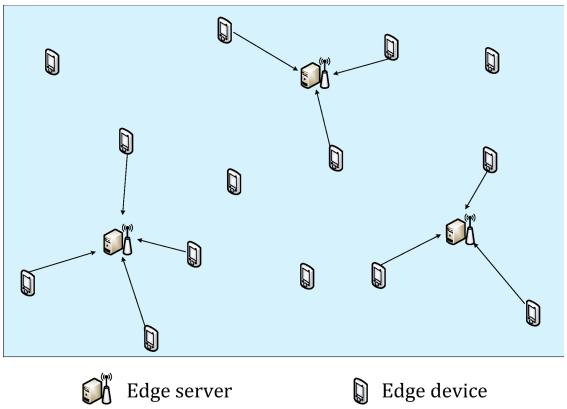

Fig. 1. Edge servers, edge devices in the network

Define γ_p as the time of maximum completion delay, which means for each task, it should be completed within γ_p, or we think this task is a failure task. Suppose edge servers are heterogeneous, and they have different calculation capabilities. In this paper, we consider the impact of transmission interference on task offloading. Therefore, in order to reduce the delay of task transmission to the edge server, we use CB and SIC technique. Define T_p as the time consumed by task $v_{pi}[k]$. We have

$$T_p = T_u[p] + T_s[p] + T_c[p], \tag{1}$$

where $T_u[p]$ is the transmission time, $T_s[p]$ is the waiting time in task queue and $T_c[p]$ is the calculation time. In the following we will give these three times in order. We will first give some symbols for preparation, then give the expressions of these times.

Define $e_p(f)$ to indicate whether task $v_{pi}[k]$ is uploaded to a server, i.e.

$$e_p(f) = \begin{cases} 1: \text{if task } v_{pi}[k] \text{ has been uploaded to server in time slot } t_f; \\ 0: \text{otherwise.} \end{cases}$$

There are two steps for a transmission when using the CB technique. First, an edge device broadcasts its data to the surrounding devices. Second, multiple devices cooperate to transmit tasks to the server by CB technique. Define $x_p(f)$ and $y_p(f)$ to indicate the transmission states, then we have

$$x_p(f) = \begin{cases} 1: \text{if the task } v_{pi}[k] \text{ is broadcasting at time slot } t_f; \\ 0: \text{otherwise.} \end{cases}$$

$$y_p(f) = \begin{cases} 1: \text{if the task } v_{pi}[k] \text{ is transmitting to an server at time slot } t_f; \\ 0: \text{otherwise.} \end{cases}$$

Obviously, task $v_{pi}[k]$ can only be in one transmission state in the same time slot. For the convenience of representation, we think $x_i(f) = 0$ when not using CB technique. Therefore, we have

$$x_p(f) + y_p(f) \leqslant 1 \quad (1 \leqslant f \leqslant h). \tag{2}$$

When using the SIC technique, edge servers may receive signals at the same time if the SINR requirement is satisfied. Define β as the SINR threshold. When task $v_{pi}[k]$ is uploaded to edge sever s_j, we have

$$SINR^k_{p\to s_j} \cdot y_p(f) = \frac{P_{p,s_j} \cdot y_p(f)}{N_0 + \sum\limits_{P_p \neq P_q}^{p \geq q} P_{q,s_j} \cdot y_q(f)} \geqslant \beta \cdot y_p(f), \tag{3}$$

where N_0 is the noise power and P_{p,s_j} is the transmission power of the task $v_{pi}[k]$ to s_j. For P_{p,s_j}, if CB technique is not used for transmitting, we have $P_{p,s_j} = (h_{i,s_j})^2 P$, and if CB technique is used, we have $P_{p,s_j} = (\sum_{d_g \in D_i(f)} h_{g,j})^2 P$, where $(h_{g,s_j})^2$ is the channel gain between the edge device d_g and the edge server s_j, and $D_i(f)$ is the set of edge devices assisting the transmission of device d_i by CB technique at time slot t_f.

When d_i broadcasts task $v_{pi}[k]$ to the surrounding devices for preparing the CB transmission, it also needs to meet the SINR requirement. Otherwise, there will be interference and the broadcasting can not be carried out smoothly. So we have

$$SINR^k_{p\to D_i(f)} \cdot x_p(f) = \frac{(h_{p,d_g})^2 P \cdot x_p(f)}{N_0 + \sum\limits_{P_p \neq P_l}^{p \geq l} (h_{l,d_g})^2 P \cdot x_l(f) + \sum\limits_{P_p \neq P_q}^{p \geq q} P_{q,d_g} \cdot y_q(f)} \tag{4}$$
$$\geqslant \beta \cdot x_p(f) \quad (d_g \in D_i(f)).$$

According to Shannon formula, we can get the broadcast transmission rate and the CB transmission rate. We have

$$r_{p\to D_i(f)} = r_{p\to s_j} = Wlog_2(1+\beta). \tag{5}$$

Now we give the formula of the first time $T_u[p]$. Notice that when $v_{pi}[k]$ is being uploaded and if CB technique is used, the transmission time can be divided into three parts. One, waiting time $t_w[p]$ for other's broadcasting or CB transmitting. Two, broadcasting time $t_b[p]$. Three, CB transmitting time $t_c[p]$; Define $\mathcal{R}_p$ as the amount of data for the task $v_{pi}[k]$. For $t_b[p]$ and $t_c[p]$, we have

$$\begin{cases} t_b[p] = \frac{\mathcal{R}_p}{r_{p\to D_i(f)}}, \\ t_c[p] = \frac{\mathcal{R}_p}{r_{p\to s_j}}. \end{cases} \tag{6}$$

For $t_w[p]$, we can further divide it into two parts. The first part is the waiting time for other's broadcasting. The second part is the waiting time for other's CB transmitting. So we have

$$t_w[p] = \sum_{l=0}^{h}((1-x_p(l))\cdot(1-e_p(l)))\cdot\tau + \sum_{l=0}^{h}((1-y_p(l))\cdot(1-e_p(l)))\cdot\tau, \tag{7}$$

where τ is the length of a time slot. Then we can express $T_u[p]$ as

$$T_u[p] = t_b[p] + t_c[p] + t_w[p]. \tag{8}$$

For the second time $T_s[p]$, we know after a new task $v_{pi}[k]$ arrives at the server, it will be put into the server's task queue. The position in the queue is determined according to the priority of the task. Define $z_{p,j}(f)$ as

$$z_{p,j}(f) = \begin{cases} 1 : f \geqslant \frac{T_u[p]}{t} + k \\ 0 : \text{otherwise}. \end{cases}$$

Apparently if $z_{p,j}(f) = 1$, it means that task $v_{pi}[k]$ has arrived to s_j at t_f.

The waiting time $T_w[p]$ depends on $v_{pi}[k]$'s position in the task queue. The higher the position in the queue, the shorter the waiting time. We set a priority variable $\omega_p(f)$ for task $v_{pi}[k]$ in time slot t_f. The priority is affected by the remaining completion time. So in different time slots, the value of priority is different. When the priority of task is higher, its position in the queue is higher. At the same time, tasks that have arrived at edge server and have been completed before the time slot t_f must not exist in the queue. Define $fun_j(\mathcal{D}_p)$ as a function of the time taken by the task $v_{pi}[k]$ to calculate on s_j. Define $\mathcal{D}_l$ as calculation amount. Therefore, for the waiting time $T_w[p]$ of task $v_{pi}[k]$ in the queue, we have

$$T_s[p] = \sum_{\omega_l(f)>\omega_p(f)} (\sum_{\frac{T_l}{t}+k\leqslant f} fun_j(\mathcal{D}_l)\cdot z_{p,j}(f)), \tag{9}$$

where T_l is the time taken for task $v_{li}[k]$ to complete.

Now we give the expression of the third time $T_c[p]$. $T_c[p]$ can be expressed as

$$T_c[p] = fun_j(\mathcal{D}_p). \tag{10}$$

Define γ_p as the maximum completion delay of task $v_{pi}[k]$. If the time $T_p \leq \gamma_p$, we think $v_{pi}[k]$ is handled successful. Define N_c as the set of all successful tasks. Our optimization goal is to maximize the task completion rate C. We have

$$\begin{array}{ll} \max & C \\ \text{s.t.} & C = \frac{|N_c|}{n} \\ & v_{pi}[k] \in N_c, \quad (T_p \leqslant \gamma_p) \\ & (1),(2),(3),(4),(5),(6),(7),(8),(9),(10). \end{array} \tag{11}$$

In Eq. (11), variables such as $x_p(f)$, $y_p(f)$, $e_p(f)$ and $z_{p,j}(f)$ determine the time T_p taken to complete task $v_{pi}[k]$. The value of $e_p(f)$, $z_{p,j}(f)$ is affected by $x_p(f)$ and $y_p(f)$. If the values of $x_p(f)$, $y_p(f)$ can be determined, $e_p(f)$, $z_{p,j}(f)$ can be determined, the problem can be solved. But it's very difficult. The $x_p(f)$, $y_p(f)$ of these tasks cannot be determined directly. Therefore, it is difficult to determine the value of $e_p(f)$, $z_{p,j}(f)$ in each time slot. The queue condition on the edge server is also difficult to determine. This kind of problem is NP-hard and cannot be solved directly. Therefore, we design a heuristic algorithm, hoping to get a feasible solution.

3 Scheduling Algorithm

As Eq. (11) is NP-hard and can not be solved directly, we need to take other strategies to solve this problem. If we can determine the values of $x_p(f)$, $y_p(f)$, $e_p(f)$ and $z_{p,j}(f)$, the model can be solved. However, it is very difficult to get the values of these variables. We have three problems: (1) Which task is calculated on which server? (2) When is the task transferred? (3) How long does it take for a task to arrive at the server?

To solve these problems, variables can be determined. Therefore, we design a heuristic algorithm to solve these problem and determine variables in the Eq. (11), so as to get the solution of the model. Through the analysis of the model, we divide the whole algorithm into three steps: (1) Edge server selection; (2) Task transmission; (3) Build task queue. Next, we will introduce these three steps in detail.

3.1 Edge Server Selection

We now discuss the Edge Server Selection algorithm. Since there are multiple edge servers in the whole network, we need to select a suitable server for each task to perform its calculation. We first define a variable $\alpha_{pj}(f)$ to indicate the fitness of the edge server s_j for the task $v_{pi}[k]$. We have

$$\alpha_{pj}(f) = \frac{\gamma_p(f) - \sum_{v_{li}[k] \in M_j(f)} T_l}{T_c[p]}, \tag{12}$$

where $M_j(f)$ is the set of tasks on the s_j at t_f, $\gamma_p(f)$ is the remaining completion time of $v_{pi}[k]$ at t_f.

We will calculate $\alpha_{pj}(f)$ in each time slot for all tasks which have not yet transmitted to servers, and then select the most fitness server based on the following rules.

One, confirm that the selected edge server is in the transmission range of the now considered edge device. Notice that in our network, even when using CB technique, for some edge devices, some servers may not be reached. This transmissions should be excluded first.

Two, $\alpha_{pj}(f) > \alpha$, where α is a fitness threshold.

Three, if for a task there exists some servers can be reached directly, i.e., not using CB technique, we will selected the closest one for this task. Otherwise, we will select the one with the largest $\alpha_j(f)$ value for this task. Since comparing with direct transmitting, CB technique will cause more interference and will lead a complexity transmission. So we have this rule.

According to the above three rules, we can select the target edge server for each task. We have Algorithm 1.

Algorithm 1: Edge Server Selection

```
Input: the set of task $V$, the set of server $S$, fitness threshold $\alpha$,
       $M_j(f) (1 \leqslant j \leqslant m)$.
Output: Task's corresponding edge server $E[v, s]$
1  for every server $s_j$ in $S$ do
2      for every task $v_{li}[k]$ in $M_j(f)$ do
3          get $\sum_{v_{li}[k] \in M_j(f)} T_l$
4      end
5  end
6  for every task $v_{pi}[k]$ in $V$ do
7      $flag1 \leftarrow 0, flag2 \leftarrow 0$;
8      if the task needs to select a server then
9          for every server $s_j$ in single-hop range do
10             find a server with biggest $\alpha_{pj}(f)$ and $\alpha_{pj}(f) > \alpha$;
11         end
12     end
13     if there is no suitable server in single-hop range then
14         for every server $s_j$ not in single-hop range do
15             find a nearest server with $\alpha_{pj}(f) > \alpha$;
16         end
17     end
18     if flag1=0,flag2=0 then
19         find nearest server $s_j$; $E[v, s] \leftarrow v_{pi}[k], s_j$;
20     end
21 end
```

3.2 Task Transmission

For the first step, we have confirmed edge servers for all tasks at current time slot. However, even using SIC technique, we can not realize all tasks be transmitted without interference. So in this step, we will decide which tasks will really be transmitted at current time slot, i.e., the Task Transmission algorithm. The main step of the Task Transmission algorithm is as following.

One, based on the remaining completion time $\gamma_p(f)$, sort all tasks from the smallest to the largest and get a task queue.

Two, check the task queue one by one, and based on SIC technique, decide the first transmitted task for each server. We give some explanations. Notice that the first task in the queue can be decided directly. Then for the second task, there may have three situations. First, the selected server is the same with the first one, then we skip this task and continue to check the following tasks. Two, the selected server is not the same with the first one, but this task will interference the first one even when using SIC, i.e., Eq. (3) or (4) will not be satisfied after adding this task, then we skip this task and continue to check the following tasks. Three, the selected server is not the same and this task will not interference with the first one, then we can decide this task. We will do it until all tasks in the queue have been checked.

Three, check the task queue again, and decide more transmitted tasks. In this step we will try to check the tasks in the queue which have not be decided again, and check if Eq. (3) or (4) can be satisfied when the task is added. If it can be satisfied, we will decide the task.

The detail steps can be found in Algorithm 2.

```
Algorithm 2: Task Transmission
   Input: Task's corresponding edge server $E[v,s]$, the set of task $V$, the set
          of edge server $S$
   Output: Updated task status
 1 for every unfinished transfer task $v_{pi}[k]$ do
 2     find a task $v_{pi}[k]$ with the least $\gamma_p(f)$;
 3     find the corresponding server $s_j$ in E[v,s];
 4     transfer task $v_{pi}[k]$ to $s_j$;
 5     for every server in $S$ do
 6         if No tasks are transferred to the server $s_j$ then
 7             find a task $v_{pi}[k]$ with the least $\gamma_p(f)$;
 8             $v_{pi}[k]$,$s_j$ in $E[v,s]$;
 9             If there is no interference, transfer the task;
10         end
11     end
12 end
13 for every unfinished transfer task $v_{pi}[k]$ do
14     find the corresponding server in E[v,s];
15     If there is no interference, transfer the task;
16 end
```

3.3 Build Task Queue

Each time when a new task reaches to a server, instead of putting the task to the tail of the task queue directly, we want to give an algorithm to decide the suitable position in the queue, i.e., the Task Queue algorithm. To do that, denote a priority value $\omega_p(f)$ for task $v_{pi}[k]$ at time slot t_f, we have

$$\omega_p(f) = \frac{T_c[p]}{\gamma_p(f)}. \tag{13}$$

We will calculate each $\omega_p(f)$, and sort them from the largest to the smallest, then get a new queue. The detail steps can be seen in Algorithm 3.

Algorithm 3: Task queue update

```
Input: the set of task $V$, the set of edge server $S$, $M_j(f)(1 \leqslant j \leqslant m)$
Output: Task queue after update
1  for every server in $S$ do
2      find the first task $v_{li}[k]$ in task queue;
3      calculates the first task $v_{li}[k]$;
4      if the $v_{li}[k]$ is completed then
5          remove this task from the task queue; $M_j(f) \rightarrow v_{li}[k]$;
6      end
7  end
8  for every server in $S$ do
9      for every task $v_{pi}[k]$ arrived at the server do
10         $\omega_p(f) = \frac{T_p}{\gamma_p(f)}$;
11         find the right location according to $\omega_p(f)$;
12         insert task $v_{pi}[k]$ into task queue; $M_j(f) \leftarrow v_{pi}[k]$;
13     end
14 end
```

4 Simulation Result

In this section, we give simulation results. Consider 3 edge servers and 20 edge devices deployed randomly in a 1000 m × 1000 m square area. Edge devices generate tasks randomly at different time slots. The whole time T is divided into 100 time slots. For these 3 edge servers, we set the processing speed is 3 GHz, 4 GHz and 5 GHz, respectively. For edge devices, we set transmission power $P = 1$ W. For tasks, we set the range of data amount $\mathcal{R}_p$ from 1 and 3 MB, and the maximum completion delay γ_p from 25 and 30 timeslots. We set $N_0 = 10^{-15}$W, $\beta = 1$ and $W = 1$ GHz. In the following we will first analyze the influence of fitness threshold α on the experimental results, and then compare our CB-SIC PRO scheduling scheme with CB-FIFO PRO, SIC PRO and SIC FIFO scheme in the same environment.

4.1 The Effect of Fitness Threshold α

In order to show the effect of the threshold α to the task completion rate and completion delay in different environments, we adjust the generated number of tasks randomly in the network. The number of tasks was 40, 50, 60 and 70 respectively. We first carry out the experiment according to CB-SIC PRO scheme. The experimental results are shown in the Fig. 2.

We can see that the change of α has a significant impact on the experimental results. In Fig. 2(a), we show the change of task completion rate under different α. We can see that when the number of tasks is 40, with the gradual increase of α, the task completion rate first rises slowly, then gradually decreases, and finally remains unchanged. When the $\alpha = 0$, the task completion rate is 98.2%. When the $\alpha = 14$, the task completion rate reaches 100%. It begins to decrease when the $\alpha = 22$. When the task completion rate drops to 98.2%, it remains unchanged. When the task number is 40, the completion rate remains at a high level, so the effect of α is not very obvious. With the increase of the task number, we can see that the fitness threshold α has a great impact on the task completion rate. Especially when the task number is 60, the lowest task completion rate is 79.1% while the highest is 94.8%. There is a 15.7% gap in task completion rate.

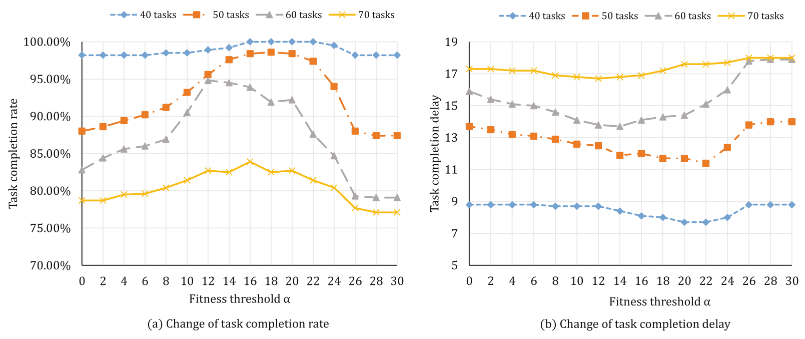

Fig. 2. The effect of the threshold α on the CB-SIC PRO scheme

We can also find that although the completion delay has a similar performance from Fig. 2(b). No matter how the number of tasks in the network changes, the task completion delay will first decrease, then slowly increase and finally remain unchanged with the increase of the threshold.

4.2 Comparison of Experimental Results

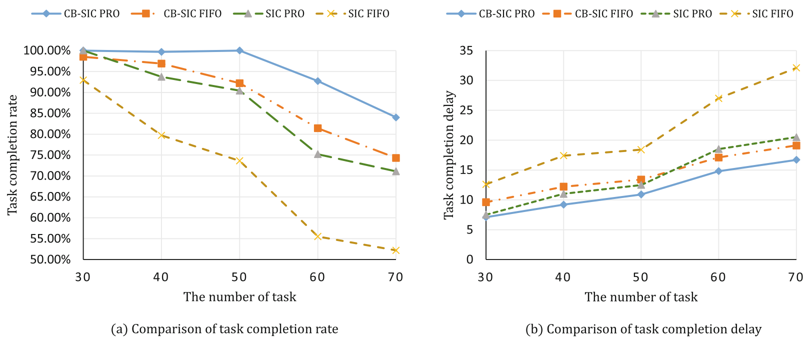

Fig. 3. Comparison of task completion rate

In Sect. 4.1, we find that when the fitness threshold $\alpha = 18$, CB-SIC PRO scheme can achieve good task completion rate and delay. Therefore, we set $\alpha = 18$ to carry out the following experiments. We generated the number of tasks randomly from 30 to 70, with the step 10. We get the change of task completion rate with different task number, and compare it with other schemes including CB-SIC FIFO, SIC FIFO and SIC PRO. The experimental results are shown in the Fig. 3.

In Fig. 3(a), we compare the task completion rate. We find that with the increase of the task number, the completion rate of all programs gradually begins to decline. But the task completion rate of CB-SIC PRO scheme is much higher than that of other schemes, and the decrease is the lowest. In Fig. 3(b), we compare the task completion delays. As the task number increases, the task completion delay will increase. However, CB-SIC PRO scheme has the lowest growth rate. It's task completion delay is always lower than the other three scheme.

Compared with the three comparative experiments, the task completion rate of our proposed scheme is much higher than other schemes, and the average task completion delay is also lower than other schemes. And with the increase of the number of tasks, the gap will become more and more obvious.

5 Conclusion

In this paper, a priority scheduling scheme is designed to improve the task completion rate for edge computing based on CB and SIC technique. When the computing resources of the nearest edge server are insufficient, edge devices can directly transfer tasks to remote idle edge server for computing. We first build a mathematical model according to the network structure. However, this model is NP-hard, which is difficult to solve directly. Therefore, we analyze the model and design a heuristic algorithm. In order to enable the task to be unloaded to a suitable edge server, we set the fitness threshold α and calculate the value of $\alpha_{pj}(f)$. We sort the task queue in edge server according to the task priority. In simulation experiments, we first show and analyze the influence of fitness threshold α. The results show that the task completion rate and completion delay will change with the change of α. Then the task completion rate and completion delay are compared. The results show that compared with other schemes, CB-SIC PRO scheme significantly improves the task completion rate and reduces the task completion delay.

References

1. Yu, L., Cai, Z.: Dynamic scaling of virtual clusters with bandwidth guarantee in cloud datacenters. In: IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pp. 1–9 (2016)
2. Xia, X., Chen, F., He, Q., Grundy, J.C., Abdelrazek, M., Jin, H.: Cost-effective app data distribution in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(1), 31–44 (2021)
3. Liu, Y., Li, Y., Niu, Y., Jin, D.: Joint optimization of path planning and resource allocation in mobile edge computing. IEEE Trans. Mobile Comput. **19**(9), 2129–2144 (2020)
4. Lin, L., Liao, X., Jin, H., Li, P.: Computation offloading toward edge computing. Proc. IEEE **107**(8), 1584–1607 (2019)
5. Dolui, K., Datta, S.K.: Comparison of edge computing implementations: fog computing, cloudlet and mobile edge computing. In: 2017 Global Internet of Things Summit (GIoTS), pp. 1–6 (2017)
6. Charyyev, B., Arslan, E., Gunes, M.H.: Latency comparison of cloud datacenters and edge servers. In: GLOBECOM 2020–2020 IEEE Global Communications Conference, pp. 1–6 (2020)
7. Wei, X., et al.: MVR: an architecture for computation offloading in mobile edge computing. In: 2017 IEEE International Conference on Edge Computing (EDGE), pp. 232–235 (2017)
8. Cai, Z., Shi, T.: Distributed query processing in the edge assisted IoT data monitoring system. IEEE Internet Things J. **7**(9), 1–1 (2020)
9. Zhu, T., Shi, T., Li, J., Cai, Z., Zhou, X.: Task scheduling in deadline-aware mobile edge computing systems. IEEE Internet Things J. **6**(3), 4854–4866 (2019)
10. Ding, Y., Liu, C., Li, K., Tang, Z., Li, K.: Task offloading and service migration strategies for user equipments with mobility consideration in mobile edge computing. In: 2019 IEEE International Conference on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 176–183 (2019)
11. Schäfer, D., Edinger, J., Breitbach, M., Becker, C.: Workload partitioning and task migration to reduce response times in heterogeneous computing environments. In: 2018 27th International Conference on Computer Communication and Networks (ICCCN), pp. 1–11 (2018)
12. Liu, Y., Peng, M., Shou, G., Chen, Y., Chen, S.: Toward edge intelligence: multi-access edge computing for 5g and internet of things. IEEE Internet Things J. **7**(8), 6722–6747 (2020)

13. Scherber, D., et al.: Coherent distributed techniques for tactical radio networks: enabling long range communications with reduced size, weight, power and cost. In: MILCOM 2013–2013 IEEE Military Communications Conference, pp. 655–660 (2013)
14. Marques, A.G., Wang, X., Giannakis, G.B.: Minimizing transmit power for coherent communications in wireless sensor networks with finite-rate feedback. IEEE Trans. Sig. Process. **56**(9), 4446–4457 (2008)
15. Shi, Y., Sagduyu, Y.E.: Coherent communications in self-organizing networks with distributed beamforming. IEEE Trans. Veh. Technol. **69**(1), 760–770 (2020)
16. Kong, J., Dagefu, F.T., Sadler, B.M.: Coverage analysis of distributed beamforming with random phase offsets using Ginibre point process. IEEE Access **8**, 134351–134362 (2020)
17. Liu, R., Shi, Y., Lui, K., Sheng, M., Wang, Y., Li, Y.: Bandwidth-aware high-throughput routing with successive interference cancelation in multihop wireless networks. IEEE Trans. Veh. Technol. **64**(12), 5866–5877 (2015)
18. Jiang, C., et al.: Cross-layer optimization for multi-hop wireless networks with successive interference cancellation. IEEE Trans. Wirel. Commun. **15**(8), 5819–5831 (2016)
19. Liang, Y., Wei, Z., Chen, Q., Wu, H.: Neighbor discovery algorithm in wireless ad hoc networks based on successive interference cancellation technology. In: 2020 International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1137–1141 (2020)
20. Adhikari, M., Mukherjee, M., Srirama, S.N.: DPTO: a deadline and priority-aware task offloading in fog computing framework leveraging multilevel feedback queueing. IEEE Internet Things J. **7**(7), 5773–5782 (2020)