A Greedy Algorithm for Task Offloading in Mobile Edge Computing System

Feng Wei[1], Sixuan Chen[1], Weixia Zou[1,2,*]

[1] Key Lab. of Universal Wireless Communications, MOE, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2] State Key Lab. of Millimeter Waves, Southeast University, Nanjing 210096, China
* The corresponding author, email: zwx0218@bupt.edu.cn

Abstract: Mobile edge computing (MEC) is a novel technique that can reduce mobiles' computational burden by tasks offloading, which emerges as a promising paradigm to provide computing capabilities in close proximity to mobile users. In this paper, we will study the scenario where multiple mobiles upload tasks to a MEC server in a sing cell, and allocating the limited server resources and wireless channels between mobiles becomes a challenge. We formulate the optimization problem for the energy saved on mobiles with the tasks being dividable, and utilize a greedy choice to solve the problem. A Select Maximum Saved Energy First (SMSEF) algorithm is proposed to realize the solving process. We examined the saved energy at different number of nodes and channels, and the results show that the proposed scheme can effectively help mobiles to save energy in the MEC system.

Keywords: mobile edge computing; task offloading; greedy choice; energy; resource allocation

I. INTRODUCTION

With the prosperity of online game, augmented reality and Internet of things (IoT), a large number of computing tasks have been generated on resource-limited devices. Mobile edge computing (MEC) can help improve the users' experiences, which provides IT and cloud-computing capabilities within the Radio Access Network (RAN) in close proximity to mobile users[1]. The cloud computing which provides virtually unlimited available resources[2] seems to be inefficient in delay-constrained applications[3]. MEC can be regard as a product of the remote cloud[4] approaching mobile users, which may benefit from the Internet architecture evolution from TCP/IP to Named Data Networking(NDN)[5][6]. Similar technologies include fog computing[7][8] and Cloudlet[9]. Fog is a "cloud close to the ground", which extends the cloud computing to the edge of networks[7]. Cloudlet complements the cloud computing model as a middle tier of 3-tier hierarchy between mobile devices and the cloud[3][9]. MEC is proposed to overcome the limitation of coverage and scalability in Cloudlet[10], which is also a complementary for cloud computing model[3]. Multi-access edge computing[11] is an evolution of MEC to coordinate different networks of 5G operators.

Computation offloading is an important use case in MEC. [12] introduced an implementation of an offloading framework in MEC architecture with an augmented reality app. Field trial results in 5G network can be found in [13]. Different optimal models were built for

Received: Nov. 15, 2017
Revised: Mar. 8, 2018
Editor: Da Chen

In this paper, the authors proposed SMSEF algorithm to reduce mobiles' energy consumption when the tasks being dividable. The algorithm utilizes a greedy choice to solve the optimization problem.

different scenarios. [14] studied an energy-efficient computation offloading (EECO) mechanism in 5G heterogeneous networks with a small BS and a macro BS sharing partial channels, and their priority-based method reduced the complexity of the optimal problem. [15] put forward a cooperative video processing scheme in multimedia IoT system. Stochastic and statistic methods are used in [16] and [17] to analyze systems' performance. [16] adopted a Markov decision process to determine where to execute a task in a single user system, and [17] assumed the arrival of the tasks as a Poisson process to maximize the profit of the edge cloud in a system with remote cloud. In a one-hop multi-user scenario, another problem is how to allocate radio resources. [18] formulated the optimization problems in TDMA and FDMA systems. [19] proposed a concept on opportunity consumption to solve the offloading problem with CSMA as its communication model. Both [20] and [21] put forward schemes in OFDMA-based cellular networks, and general constraints for channel allocation can be found in [14]. Besides, a multi-cell scenario was considered in [22], in which the inter-cell interference and finite computation resources limited the scalability of offloading. The MEC model for 5G in [23] is suitable for multi-cell scenario.

The priority-based algorithm in [14] is impressive, but it deals with un-dividable tasks. [18] involves dividable tasks, but the analysis of MEC processing, i.e. queuing and computing, is not detailed as that of their channel allocation and bits division.

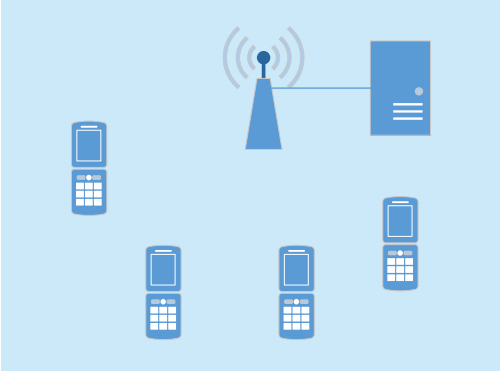

Fig. 1. *A mobile edge computing system in a cell.*

In this paper, we will investigate an MEC system in which mobile users offload their computation tasks to one MEC server through one-hop wireless channels. As the shortage of computing and channel resources, the MEC server needs a resource allocation scheme to coordinating the competition between mobiles. We firstly formulate the optimization problem to maximize the energy saved in mobiles with tasks being dividable. Then we utilize greedy choice to decompose the optimization problem after observing the optimal structure and subproblems' similarity. The queuing time is a product of mobiles' competition, and the decomposition avoids the analysis of queuing time, reducing the complexity of the optimization problem. We adopt the maximum saved energy for each greedy choice and propose a Select Maximum Saved Energy First (SMSEF) algorithm. Simulation results show that the proposed scheme can effectively help mobiles to save energy in the MEC system.

The rest of this paper is organized as follows. We will introduce the communication model and computation model for the MEC system in Section 2. In Section 3, an optimization problem is formulated to maximize the energy saved in mobiles at first. Then we provide a greedy method to decompose the problem. At the end of Section 3, we propose the SMSEF algorithm in detail. Section 4 is the simulation results.

II. Model

Let us consider a mobile edge computing (MEC) system in figure 1 with N mobiles in the same cell within one base station (BS). The mobile set is denoted by $\mathbf{N}=\{1,2,\ldots,N\}$ and every mobile in $\mathbf{N}$ has a task to be computed. Each task is characterized by three terms $\text{task}_i=(w_i, a_i, t_i)$, $i\in\mathbf{N}$. w_i is the total CPU cycle to accomplish the task, a_i is the amount of input data and t_i is the maximum latency of the task. The divided task with proportion q_i of task i is (q_iw_i, q_ia_i, t_i). Any divided task can be computed locally or remotely. When uploaded to the MEC server, a task will take up a period

in MEC task queue and be computed sequentially, which is illustrated in figure 2. We will analyze the model as follows.

2.1 Communication model

The spectrum is divided into K channels, which are denoted as $\mathbf{K}=\{1,2,\ldots,K\}$. The bandwidth of each channel is B and it is orthogonal to the others. Suppose the channel selection indicator of mobile i in channel k at time slot t is c_{ikt}, i.e. channel selection indicator matrix $\mathbf{C}=\{c_{ikt}\}$, $i\in\mathbf{N}$, $k\in\mathbf{K}$, $t\in\{1,2,3,\ldots\}$, where

$$c_{ikt}=\begin{cases}1 \text{ if } i \text{ occupy channel } k \text{ at slot } t\\ 0 \qquad\qquad\qquad \text{else}\end{cases} \tag{1}$$

For a given i and t $\mathbf{c}_{it}=(c_1,c_2,\ldots,c_K)$, there is at most one element is 1, i.e. the inner product of c_{it} and the same size vector with all elements being one is less than 1, which means a user can occupy at most one channel at a given time slot.

$$\mathbf{1}\bullet\mathbf{c}_{it}\le 1 \quad \forall i\in\mathbf{N}, t\in\{1,2,3,\ldots\} \tag{2}$$

And the channel matrix is $\mathrm{G}=\{g_{ikt}\}, i\in\mathbf{N}$, $k\in\mathbf{K}$, $t\in\{1,2,3,\ldots\}$, and for a given i and t, $\mathbf{G}_{it}=(g_1,g_2,\ldots,g_K)$. Then rate in tth time slot for mobile i is as follows

$$R_{it}=B\log_2\left(1+\frac{P_{it}\mathbf{c}_{it}^T\mathbf{G}_{it}}{N_0}\right) \tag{3}$$

Then the total amount of transmitted data of task i is

$$D_i^{tx}=T_0\sum_t R_{it} \tag{4}$$

where T_0 is the time slot length.

2.2 Computation model

1) Computed Locally: The local processing time of task $(q_i^l w_i, q_i^l a_i, t_i)$ is given as

$$T_i^l=\frac{q_i^l w_i}{f_i^l} \tag{5}$$

where f_i^l is the local computation capability.

The energy consumed in mobile is as follows:

$$E_i^l=r_i q_i^l w_i \tag{6}$$

where r_i is a coefficient for energy consumed by every CPU cycle.

2) Computed Remotely: For an uploaded task $(q_i^c w_i, q_i^c a_i, t_i)$, the MEC server may divide it into multiple subtasks to adapt to the discontinuous processing period in the server, which is illustrated in figure 3.

We suppose the subtask number of task i is M_i, then a subtask can be expressed as $(q_{ij}^c w_i, q_{ij}^c a_i, t_i)$, $j=1,2,\ldots,M_i$, where $\sum_{j=1}^{M_i} q_{ij}^c=q_i^c$, and subtasks are numbered in chronological order. The processing time of task $(q_{ij}^c w_i, q_{ij}^c a_i, t_i)$ by remote computing is made up by three part: transmitting, queuing and computing. The total transmitting time for task$_i$ lasts until the beginning slot of the last computing period. The slots in which no data is sent compose the queuing period.

We divide the computing period in MEC server into slots of length T_0, and denote the ith-mobile's occupation indicator for slot t in MEC server as s_{it}. The occupation indicator matrix is $\mathbf{S}=\{s_{it}\}$, $i\in\mathbf{N}$, $t\in\{1,2,3,\ldots\}$. Every computing slot in the MEC server can be assigned to one user at most.

$$s_{it}=\begin{cases}1 \text{ if } i \text{ occupy slot } t\\ 0 \qquad\qquad \text{else}\end{cases} \tag{7}$$

The processing time for the jth subtask for

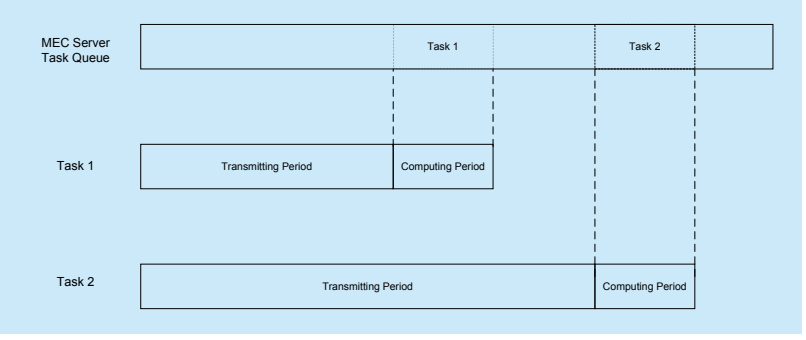

Fig. 2. *Two tasks uploaded to the MEC server.*

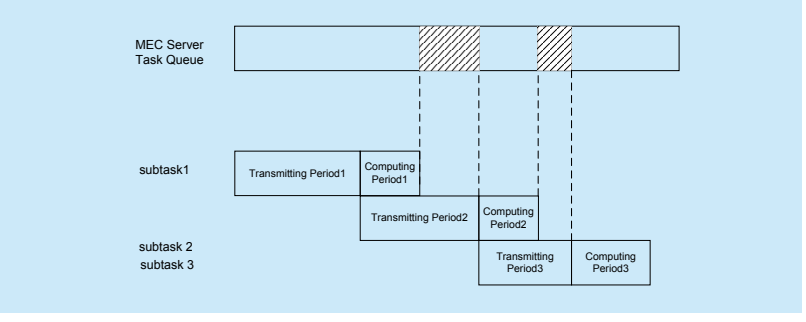

Fig. 3. *An uploaded task composed of three subtasks.*

mobile i (subtask$_{ij}$, for simplicity) is

$$T_{ij}^{p} = \frac{q_{ij}^{c} w_i}{f_i^{c}} \tag{8}$$

where f_i^c is the remote computation capability. Therefore, the remote processing time for subtask$_{ij}$ is

$$T_{ij}^{c} = \sum_{l=1}^{j} T_{il}^{x} + T_i^{p} + T_i^{q} \tag{9}$$

where T_i^q is the queuing time for jth subtask of node i and T_{il}^x is the transmitting time.

The transmitted data amount of subtask$_{ij}$ is

$$D_{ij}^{tx} = T_0 \sum_{t=t_{sum}}^{t_{sum}+T_{ij}^{x}} R_{ijt} \ , \ t_{sum} = \sum_{l=1}^{j-1} T_{il}^{x} \tag{10}$$

where R_{ijt} is the data rate in time slot t for subtask$_{ij}$. The data amount of subtask$_{ij}$ computed by the MEC server is

$$D_{ij}^{c} = q_{ij}^{c} a_i \tag{11}$$

The energy consumed in mobile is as follows:

$$E_i^{tx} = T_0 \sum_{m=1}^{T_i^x/T_0} p_{im} \tag{12}$$

where P_{im} is the transmission power in time slot m for for task i. When a task ($q_i^c w_i$, $q_i^c a_i$, t_i) is uploaded to the MEC sever, the energy saved by the mobile can be expressed as:

$$E_i = E_i^{l} - E_i^{tx} = q_i^{c} r_i w_i - T_0 \sum_{m=1}^{T_i^x/T_0} p_{im} \tag{13}$$

III. Optimization Problem and Greedy Choice

In this section, we will introduce our SMSEF algorithm. We firstly formulate an optimization problem, which maximize the energy saved by mobiles. Then we describe the optimal substructure of the problem and the similarity of the subproblems. The algorithm will be described in detail at last.

1) Optimal problem and optimal substructure

To simplify the description, we denote y_i as the strategy of mobile i to choose the combination of MEC period and wireless channel. The optimization problem is as in (14). Constraint (c1) indicates the maximum latency of a task and constraint (c2) means the transmitted data amount in the channel equal to the computed data amount in the server. Constraint (c3) keeps the causality of the process, i.e. the data should be transmitted before computed. Constraint (c6) implies the mobile can only occupy at most one channel at a time slot.

$$P1 \quad F = \sum_i f(\mathbf{y}_i) = \max \sum_i q_i^{c} r_i w_i - T_0 \sum_{m=1}^{T_i^x/T_0} p_{im}$$

$$s.t. \ T_{ij}^{c} \le t_i \ \ j = 1,...,M_i \qquad (c1)$$

$$D_i^{tx} = D_i^{c} \qquad (c2)$$

$$\sum_{j=1}^{m} D_{ij}^{tx} \ge \sum_{j=1}^{m} D_{ij}^{c} \ \ m = 1,...M_i - 1 \qquad (c3)$$

$$R_{ijt} = B \log_2 (1 + \frac{P_{ijt} c_{it}^{T} G_{it}}{N_0}) \qquad (c4)$$

$$D_{ij}^{tx} = T_0 \sum_{t=t_{sum}}^{t_{sum}+T_{ij}^{x}} R_{ijt} \ , \ t_{sum} = \sum_{l=1}^{j-1} T_{il}^{x} \qquad (c5)$$

$$\mathbf{1} \bullet c_{it} \le 1 \ \ \forall i \in \mathbf{N}, t \in \{1,2,3,...\} \qquad (c6)$$

$$D_{ij}^{c} = q_{ij}^{c} a_i \qquad (c7)$$

(14)

where, y_i is the ith user's strategy which consists of occupied channels and occupied MEC computing slots. f is a mapping from user strategy to saved energy.

$$\mathbf{y}_i = \{[\mathbf{c}_{it}, \mathbf{s}_{it}]\}, \ t \in \{1,2,3,...\} \tag{15}$$

In the following, we will show the optimal substructure of problem (14). For simplicity, we denote a MEC system with its unoccupied channel and computing occupation indicator matrices Y=[C, S].

We denote the optimal y^* as the optimal occupancy vector for system [C, S], and $y^*=[y^*_1,\ldots, y^*_N]$. When removing the solution for task i in the optimal occupancy vector, i.e. removing $y_i^*=\{[\mathbf{c}_{it}^*, \mathbf{s}_{it}^*]\}$ in y^*, the remaining part $y^*_{N\backslash i}=[y^*_1,\ldots, y^*_{i-1}, y^*_{i+1},\ldots, y^*_N]$ is an optimal occupancy vector for system [C\{$\mathbf{c}_{it}^*$}, S\{$\mathbf{s}_{it}^*$}], which can be easily proved by cut-and-paste method. Denote F(i,Y) as the maximum energy saved by offloading i tasks to the MEC system Y=[C, S]. By optimal substructure, we can get follows:

$$F(i,Y)=\begin{cases}0 & if\ \ i=0\ or\ Y=\varnothing\\ \max\limits_{y_i}(f(y_i)+F(i-1,Y\setminus\{y_i\})) & otherwise\end{cases} \tag{16}$$

2) Greedy choice and SMSEF algorithm

Problem (14) may do not have overlapping subproblems as the occupation order affecting the result, but they are much similar. For example, when offloading task i and j to the MEC system Y_0 by two orders ij and ji respectively, the result system state Y_{ij} and Y_{ji} may be different, as the prior task will occupy the time slots only on its own benefit. However, Both Y_{ij} and Y_{ji} will occupy the channels with higher gain and occupy the MEC periods conducive to the completion of the tasks.

Considering the similarity of subproblems, we adopt the following greedy sequential selection to solve problem (14): During each iteration, we choose the task that saves the most energy on the present system, which can be expressed as follows:

$$\begin{aligned}&[\pi_i, y_{\pi_i}]=\arg\max_{(i,y_i)} f(y_i\mid Y_{i-1})\ i\in N\setminus Y_{i-1}\\ &Y_i=Y_{i-1}\cup\{y_{\pi_i}\}\end{aligned} \tag{17}$$

The sequential selection may cause subsequent tasks to adjust their strategies, which is illustrated in figure 4.

Task1 can be represented as (w_1, a_1, t_1) and task$_2$ as (w_2, a_2, t_2). In figure 4(a), the shaded period (t_s, t_e) is already occupied, but the solutions for task1 and task2 conflict with the occupied period. As the deadline slot of task1 is within the occupied period, i.e. $t_s < t_1 \le t_e$, we reduce the maximum latency of task1, and the adjusted task becomes (qw_1,qa_1,t_s), where q is the uploaded proportion. For task2 when some MEC processing slots within the period and deadline slot out of the period, we cut the task into two subtasks, (q_1w_2, q_1a_2,t_s) and (q_2w_2, q_2a_2, t_2), where q_1 and q_2 are the uploaded proportion, respectively. The transmitting time of the two subtasks cannot be overlapped as they are from the same mobile.

We adopt two policies to ensure the two constraints in problem (14). The first one is that occupying the MEC time slot start from the deadline, increasing reversely. A slot closer to the deadline slot implies more channels available and fewer potential competitors. The second one is iterative water filling multiple subtasks from the same task. We illustrate the iterative water filling with a task (w_0, a_0, t_0) of three subtasks (w_0a_i/a_0, a_i, t_i) i=1, 2, 3 in figure 3, where $a_1+a_2+a_3=a_0$.

Firstly, the three-part transmitting period is treated as a whole to support the three-part data computed in MEC in iterative water filling. The following equation holds:

$$D_1^{tx}+D_2^{tx}+D_3^{tx}=D_1^{c}+D_2^{c}+D_3^{c} \tag{18}$$

Then we sequentially check the causality of the result through the following inequalities.

$$\begin{aligned}&D_1^{tx}\ge D_1^{c} && (a)\\ &D_1^{tx}+D_2^{tx}\ge D_1^{c}+D_2^{c} && (b)\end{aligned} \tag{19}$$

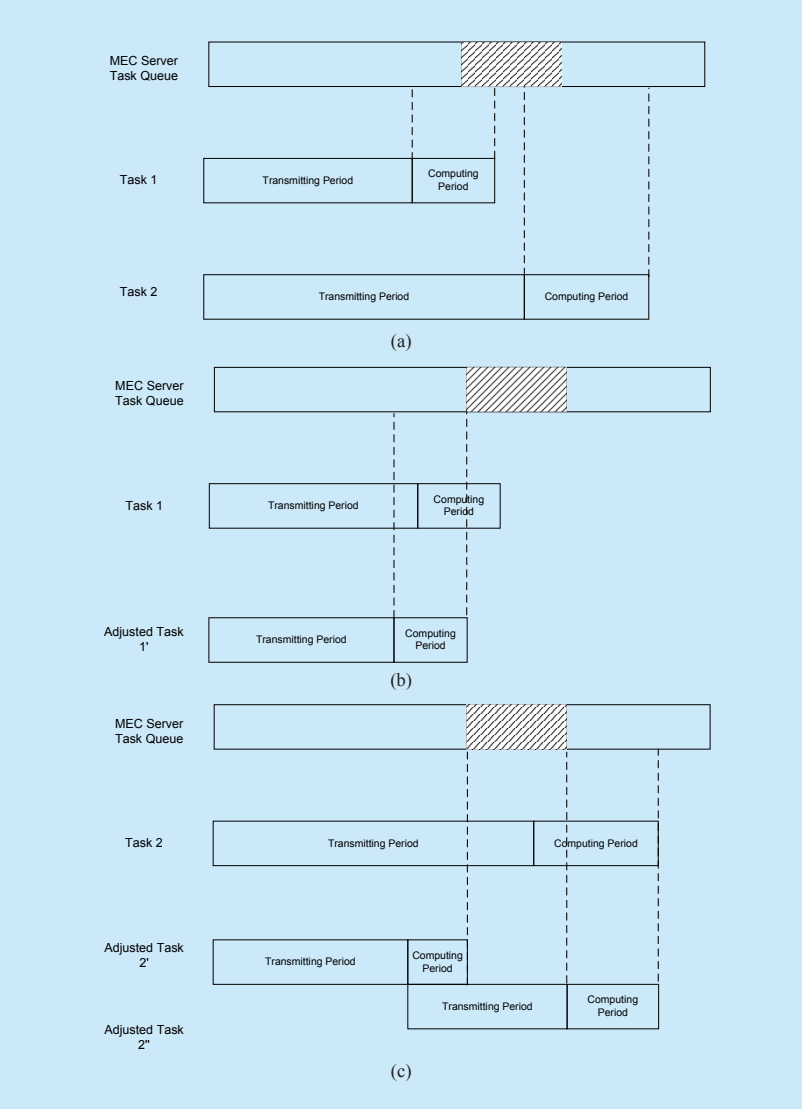

Fig. 4. *Two cases for task adjustment.*

If (a) does not hold, we separate subtask1 from subtask2 and subtask3, and water fills the two parts respectively. If (b) does not hold, we separate subtask1 and subtask2 from subtask3, and water fills the two parts respectively. It means we separate the subtasks into two parts once the subtask not satisfying the causality. When all parts satisfy the causality, the iterative water filling process is over. The SMSEF algorithm is shown in Algorithm 1. and its computation complexity is $O(N^2A)$, where A=max(a_i), i=1,…,N.

IV. Simulation

In this section, we examine the algorithm's behavior on plenty of randomly generated graphs. In the following, we will introduce simulation parameters at first, and then present the result of our simulation. The simulation settings are as follows unless specified otherwise. The local CPU computation capacity is randomly set from 0.5GHz to 1GHz and the MEC server's CPU computation capacity is 4GHz. For each task, the input data size is randomly distribute between 100KB and 1000KB, and the number of CPU cycles required by each task is randomly distribute between 200MHz and 1000MHz, and the energy consumed for every CPU cycle is 0.1uJ. The maximum latency is the duration of locally execution plus a random within 100ms .The noise power is -95dBm. The bandwidth for every channel is 5MHz. The duration of time slot is 1ms.

For comparing the performance, we consider other two algorithms as follows:

Priority-based Algorithm (Prio): This algorithm is a simplified version of EECO mechanisms in [14].The order of device i in the process of the radio resource allocation is given by a priority function:

$$Pr_i = 0.5\times(\frac{t_i^{\max}}{\bar{T}_{\max}}) + 0.5\times(\frac{\bar{E}}{E_i^l}) \quad (20)$$

where $\bar{T}_{\max} = \sum_{i\in N} t_i^{\max}$, and $\bar{E} = \sum_{i\in N} E_i^l$. The device with the less Pr_i value has a higher priority. We keep the assumption in [14] that a task cannot be divided, so it will be computed as a whole, locally or remotely.

Non-dividable SMSEF (N-SMSEF) algorithm: This algorithm is similar to the SMSEF algorithm except that a task cannot be divided, so it will be computed as a whole, locally or remotely.

Figure 5 illustrates the energy saved by different algorithms in terms of nodes' total number (i.e. the tasks' total number) when the channel number is 10. In this figure, the energy saved by the algorithms increase with the node number. The energy saved by the SMSEF algorithm is higher than the other two algorithms, especially with large number of nodes. It worth noting that there is an intersection point for N-SMSEF and Priority-based algorithm, and we will explain the reason in the following.

Table 1 and table 2 are snapshots of simulation when the node number is 6 and 15, respectively, which are sample points in figure 5. Priority-based algorithm incorporates the task's maximum latency into the priority rating. As a result, the MEC time slots assigned to each task are closer to the deadline slots. Compared to the priority algorithm, the

Algorithm 1. SMSEF Algorithm.

Input node set nodeSet, three vectors for tasks (***w***,***a***,***t***), channel matrix G, allocation state of MEC S_{MEC}, allocation state of Channels S_{CH}.
Output: allocation state of *MEC* S_{MEC}, allocation state of Channels S_{CH}, total energy saved E_{sv}.
E_{sv}=0;
unmarkedSet <- *nodeSet*
while ~isempty(*unmarkedSe*t)
for *i*=*unmarkedSet*
  optCH(*i*)<-Find the optimal channel vector for node *i*.
  For the time slots may occupied by node i, calculate the maximum value of saved energy, and iteratively water fills the channel. We get *savedEnergy*(*i*), *ocpMEC*(*i*), *ocpCH*(*i*), *Power*(*i*).
end for
chosenNode<-argmax(*savedEnergy*(*unmarkedSet*))
E_{sv} <- E_{sv} + *savedEnergy*(*chosenNode*)
update S_{MEC} by *ocpMEC*(*chosenNode*)
update S_{CH} by *ocpCH*(*i*)
remove *chosenNode* in *unmarkedSet*
end while
return S_{MEC}, S_{CH}, E_{sv}

N-SMSEF algorithm is more likely to discard the tasks with earlier deadline. When the number of nodes (i.e. the number of tasks) is small as illustrate in Table 1, the benefit from the last few items in N-SMSEF cannot compensate for the performance degradation brought by former items. In this case, the Priority-based algorithm is better than the N-SMSEF algorithm, which is the left side of the intersection point in figure 5.

N-SMSEF algorithm select tasks in the order of their objective function values, which reduces the effect of maximum latency. When the number of nodes (i.e. the number of tasks) is large, as illustrate in Table 2, the loss brought by former items is made up by the benefit from the last few items in N-SMSEF algorithm. In this case, the N-SMSEF algorithm is better than the Priority-based algorithm, which is the right side of the intersection point in figure 5.

Compared to N-SMSEF algorithm in Table1 and Table2, the SMSEF algorithm has a tail of tasks with lower uploaded proportion. In figure 6, we separate the energy saved by tasks with uploaded ratio greater than 0.85 and less than 0.85 in SMSEF algorithm and compare it with that in N-SMSEF algorithm. As illustrated in figure 6, the saved energy with uploaded proportion greater than 0.85 is smaller than that in N-SMSEF algorithm but the tasks with uploaded proportion less than 0.85 make up for this vacancy.

Figure 7 and 8 illustrate the energy saved by different algorithms in terms of channel number (i.e. the tasks number), when the node number is 6 and 15 respectively. In the figures, the energy saved by the algorithms increase as the number of channel grows. The energy saved by the SMSEF algorithm is higher than that of the other two algorithms, especially with larger number of nodes. It worth noting that the curves of N-SMSEF and Priority-based algorithm exchange their order in two figures, which coincides with the intersection point in figure 5: In figure 7, when the node number is six, the Priority-based algorithm is better than N-SMSEF algorithm, which coin-

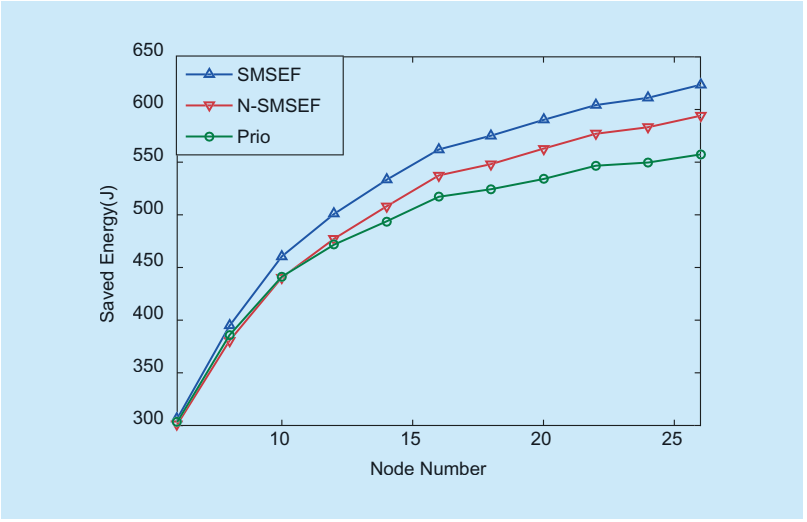

Fig. 5. *Saved Energy when channel number is 10.*

Table I. *Energy saved in simulation snapshots with 6 mobiles.*

(a) *by Priority-based Algorithm*

Node	4	6	1	5	3	sum
Saved Energy	59.9463	84.9762	77.1406	62.6866	54.6486	339.3982

(b) *by N-SMSEF Algorithm*

Node	6	1	5	3	4	sum
Saved Energy	84.9762	77.1406	62.7977	55.4334	52.4225	332.7704

(c) *by SMSEF Algorithm*

Node	6	1	5	3	4	2	sum
Proportion	1	1	1	1	0.9177	0.5	
Saved Energy	84.9762	77.1406	62.7977	55.4334	54.8042	7.0967	342.2488

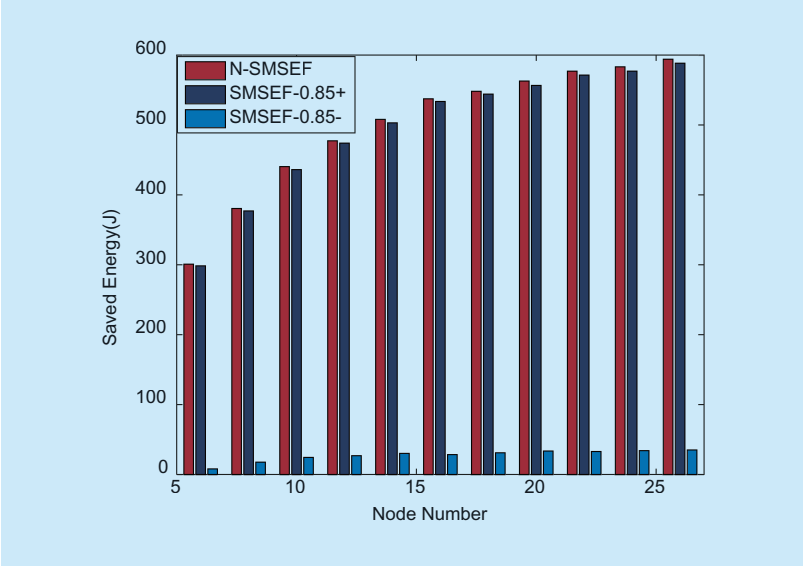

Fig. 6. *Comparison for N-SMSEF and SMSEF.*

Table II. *Energy saved in simulation snapshots with 15 mobiles.*

(a) *by Priority-based Algorithm*

Node	2	11	15	13	7	10	sum
Saved Energy	93.2380	93.4752	96.3596	79.0346	53.9217	68.8506	484.8796

(b) *by N-SMSEF Algorithm*

Node	15	11	2	13	10	5	sum
Saved Energy	96.3596	93.3342	92.9342	79.0346	73.3441	68.7113	503.7180

(c) *by SMSEF Algorithm*

Node	15	11	2	13	10	5	1
Proportion	1	1	1	1	1	1	0.4783
Saved Energy	96.3596	93.3342	92.9342	79.0346	73.3441	68.7113	28.8530
Node	3	12					sum
Proportion	0.0563	0.0345					
Saved Energy	2.9253	1.1764					537.2701

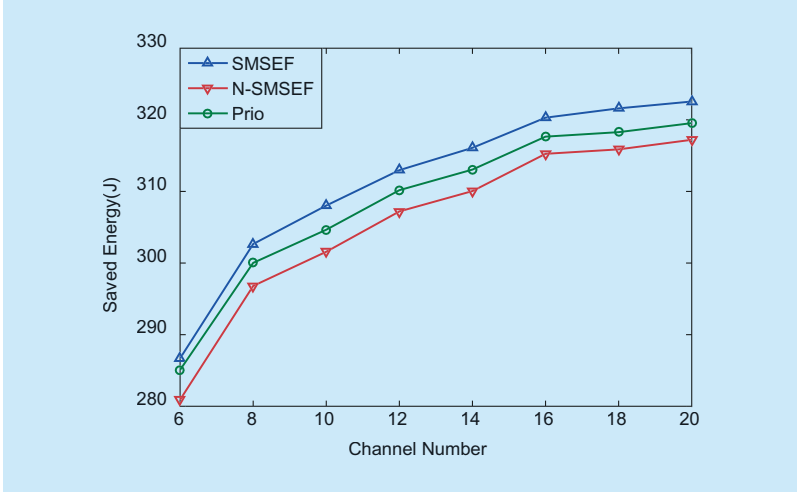

Fig. 7. *Saved Energy when node number is 6.*

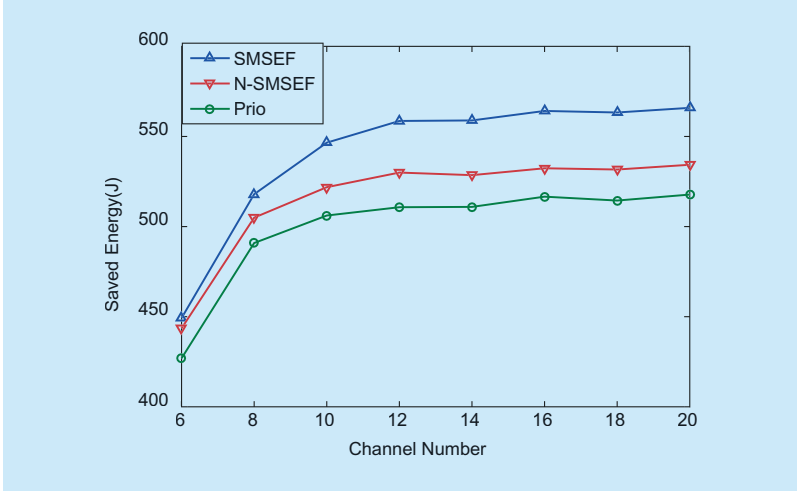

Fig. 8. *Saved Energy when node number is 15.*

cides the result of the left side of the intersection in figure 5. Similarly, Figure 8 coincides the result of the right side of the intersection in figure 5. Besides, the numerical value of each curve in figure 7 is smaller than that of the curve of the same algorithm, which coincides the increasing tendency in figure 5.

V. Conclusion

In this paper, we proposed SMSEF algorithm to reduce mobiles' energy consumption when the tasks being dividable. The algorithm utilizes a greedy choice to solve the optimization problem and producing solutions for MEC computation resource allocation, channel allocation and power control. Compared to the other two algorithms, SMSEF algorithm saves the most energy in the MEC system, and its superiority is not affected by the number of nodes or channels in given scenarios.

Acknowledgements

This work was supported by NSFC (No. 61571055), fund of SKL of MMW (No. K201815), Important National Science & Technology Specific Projects(2017ZX03001028).

References

[1] Patel M, Naughton B, Chan C, et al. Mobile-edge computing introductory technical white paper[J]. White Paper, Mobile-edge Computing (MEC) industry initiative, 2014.

[2] L. Zhao, M. Du, L. Chen. A New Multi-Resource Allocation Mechanism: A Tradeoff between Fairness and Efficiency in Cloud Computing[J]. *China Communications*, 2018, 15(3): 57-77.

[3] Ahmed E, Rehmani M H. Mobile Edge Computing: Opportunities, solutions, and challenges[J]. *Future Generation Computer Systems,* 2017 (70): 59-63.

[4] Armbrust M, Fox A, Griffith R, et al. A view of cloud computing[J]. *Communications of the ACM*, 2010, 53(4): 50-58.

[5] M. Amadeo, C. Campolo, and A. Molinaro, "NDNe: Enhancing Named Data Networking to Support Cloudification at the Edge," I*EEE Commun*. Lett., vol. 20, no. 11, pp. 2264–2267, 2016.

[6] B. Liu, T. Jiang, Z. Wang, and Y. Cao, "Object-Oriented Network: A Named-Data Architecture Toward the Future Internet," *IEEE Internet Things J.*, vol. 4, no. 4, pp. 957–967, Aug. 2017.

[7] Zhu J, Chan D S, Prabhu M S, et al. Improving web sites performance using edge servers in fog computing architecture[C]//*Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on*. IEEE, 2013: 320-323.
[8] Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the internet of things[C]//*Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. *ACM*, 2012: 13-16.
[9] Shaukat U, Ahmed E, Anwar Z, et al. Cloudlet deployment in local wireless networks: Motivation, architectures, applications, and open challenges[J]. *Journal of Network and Computer Applications*, 2016, 62: 18-40.
[10] Ahmed A, Ahmed E. A survey on mobile edge computing[C]//*Intelligent Systems and Control (ISCO)*, 2016 10th International Conference on. IEEE, 2016: 1-8.
[11] Taleb T, Samdanis K, Mada B, et al. On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Architecture & Orchestration[J]. *IEEE Communications Surveys & Tutorials*, 2017.
[12] Dolezal J, Becvar Z, Zeman T. Performance evaluation of computation offloading from mobile device to the edge of mobile network[C]//*Standards for Communications and Networking (CSCN), 2016 IEEE Conference on*. IEEE, 2016: 1-7.
[13] Zhang J, Xie W, Yang F, et al. Mobile edge computing and field trial results for 5G low latency scenario[J]. *China Communications*, 2016, 13(Supplement2): 174-182.
[14] Zhang K, Mao Y, Leng S, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. *IEEE Access*, 2016, 4: 5896-5907.
[15] Long C, Cao Y, Jiang T, et al. Edge Computing Framework for Cooperative Video Processing in Multimedia IoT Systems[J]. *IEEE Transactions on Multimedia*, 2017.
[16] Liu J, Mao Y, Zhang J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//*Information Theory (ISIT), 2016 IEEE International Symposium on*. IEEE, 2016: 1451-1455.
[17] Zhao T, Zhou S, Guo X, et al. Pricing policy and computational resource provisioning for delay-aware mobile edge computing[C]//*Communications in China (ICCC), 2016 IEEE/CIC International Conference on*. IEEE, 2016: 1-6.
[18] You C, Huang K, Chae H, et al. Energy-efficient resource allocation for mobile-edge computation offloading[J]. *IEEE Transactions on Wireless Communications*, 2017, 16(3): 1397-1411.
[19] Tianze L, Muqing W, Min Z. Consumption considered optimal scheme for task offloading in mobile edge computing[C]//*Telecommunications (ICT), 2016 23rd International Conference on*. IEEE, 2016: 1-6.
[20] Cao Y, Jiang T, Wang C. Optimal radio resource allocation for mobile task offloading in cellular networks[J]. *IEEE Network*, 2014, 28(5): 68-73.
[21] Cao Y, Long C, Jiang T, et al. Share communication and computation resources on mobile devices: a social awareness perspective[J]. *IEEE Wireless Communications*, 2016, 23(4): 52-59.
[22] Deng M, Tian H, Lyu X. Adaptive sequential offloading game for multi-cell Mobile Edge Computing[C]//*Telecommunications (ICT), 2016 23rd International Conference on*. IEEE, 2016: 1-5.
[23] Ketykó I, Kecskés L, Nemes C, et al. Multi-user computation offloading as multiple knapsack problem for 5G mobile edge computing[C]//*Networks and Communications (EuCNC), 2016 European Conference on*. IEEE, 2016: 225-229.

Biographies

Feng Wei, received the B.S. degree in Communication Engineering from Beijing University of Posts and Telecommunications, China, in 2011. He is currently working toward the Ph.D. degree at the Department of Information and Communication Engineering, Beijing University of Posts and Telecommunications. His research interests include mobile edge computing, network coding and wireless resource allocation.

Sixuan Chen, received the M.E. degree in Information and Communication Engineering from Chongqing University of Posts and Telecommunications, China, in 2013. She is currently working toward the Ph.D. degree at the Department of Information and Communication Engineering, Beijing University of Posts and Telecommunications. Her research interests include network coding and short-range wireless communications.

Weixia Zou, received her Bachelor's degree from Tongji University in 1994 and her Master's degree in Shandong University in 2002, and PhD degree from Beijing University of Posts Telecommunications (BUPT) in 2006. At present, she is an associate professor in BUPT, China. Her current research focused on new technologies of short-range wireless communications and the electromagnetic compatibility (EMC).