A Sufferage offloading tasks method for multiple edge servers

Tao Zhang[1], Mingfeng Cao[1*] and Yongsheng Hao[2]
[1]Changde City Tobacco Company, Changde, 415000, China
[2] Network Center, Nanjing University of Information Science & Technology, Nanjing, 210044, China
[E-mail : 82412379@qq.com, a13762606762@163.com, yshao@nuist.edu.cn]
*Corresponding author: Mingfeng Cao

Received December 5, 2021; revised June 30, 2022; accepted Septebmer 21, 2022; published November 30, 2022

Abstract

The offloading method is important when there are multiple mobile nodes and multiple edge servers. In the environment, those mobile nodes connect with edge servers with different bandwidths, thus taking different time and energy for offloading tasks. Considering the system load of edge servers and the attributes (the number of instructions, the size of files, deadlines, and so on) of tasks, the energy-aware offloading problem becomes difficult under our mobile edge environment (MCE). Most of the past work mainly offloads tasks by judging where the job consumes less energy. But sometimes, one task needs more energy because the preferred edge servers have been overloaded. Those methods always do not pay attention to the influence of the scheduling on the future tasks. In this paper, first, we try to execute the job locally when the job costs a lower energy consumption executed on the MD. We suppose that every task is submitted to the mobile server which has the highest bandwidth efficiency. Bandwidth efficiency is defined by the sending ratio, the receiving ratio, and their related power consumption. We sort the task in the descending order of the ratio between the energy consumption executed on the mobile server node and on the MD. Then, we give a “suffrage” definition for the energy consumption executed on different mobile servers for offloading tasks. The task selects the mobile server with the largest suffrage. Simulations show that our method reduces the execution time and the related energy consumption, while keeping a lower value in the number of uncompleted tasks.

Keywords: multiple edge servers, sufferage, offload method, tradeoff

http://doi.org/10.3837/tiis.2022.11.008 **ISSN : 1976-7277**

1. Introduction

Mobile devices bring great convenience to people's life [1]–[3] But the limitation in processing ability and energy supply hinders its usage [4]. Offloading some tasks to remote mobile servers is a solution to overcome those shortcomings [5]–[9]. The Offloading method brings two benefits for the mobile device (MD): improving the processing ability and reducing energy consumption. It makes that the MD overcomes overloaded when it has a higher system load. With the help of the remote servers, the MD also can reduce the energy consumption in some cases.

Several prototype systems have been used to offload tasks for MDs [1]–[3], [10]–[12] with different targets and different environments. The target of offloading tasks from local to a server includes: reducing the energy consumption of the MD, shortening the execution time, and meeting other QoSs (Quality of Services) [13]–[15]. If the execution location is select smartly, energy consumption can be reduced while keeping other requirements [16], [17]. How select the execution location is the key problem for offloading tasks for MDs. The network also influences the energy consumption for offloading tasks, because the different network has different bandwidth and energy consumption, which influences the execution time for transferring files and the related energy consumption. The dynamic network makes whether offloading has a different effect on the MD of the energy consumption, execution time, and so on. Prior work has been widely evaluated in various networks, such as 2G/3G/4G/5G network [7], [8], [18]–[20].

Previous offloading methods always begin with judging the location where the task has lower energy consumption [2], [21]. Then, according to the deadline of tasks, the bandwidth between the MD and the offloaded target location, the system load of the MD and remote server, the scheduling method selects the execution location [8], [17], [22]. They always judge the preferring location of a task and offload tasks according to the preferring locations. Different routes have various energy efficiencies (sending power consumption/sending rate, or receiving power consumption/receiving rate). This is the first aspect that we try to pay attention to in this paper. If we can offload to many mobile servers make the offloading problem more difficult. Moreover, the past methods always neglect the fact that the task influences each other. For example, if a task executed on the MD consumes lower energy than in the mobile cloud (2 energy on the MD and 3 energy on the offloaded location), but the next task (1 energy on the MD and 10 energy on the offloaded location) may not be executed on the MD because of the system load. The condition happens more usually in the many (mobile nodes)-to-many (mobile servers) environment. In this paper, we try to offload tasks in a many-to-many environment by considering the scheduling influence of tasks on each other.

We organize the paper as follows. Section 2 reviews the offloading methods in mobile environment. In Section 3, we give the system architecture and related models used in our paper. Section 4 gives a deep analysis of our system, and then gives the offloading method. Section 5 compares our proposed method with other methods. Section 6 gives the conclusion and the future work.

2. Related work

Offloading methods help MDs to enhance processing ability and lengthen the working time. Researchers have much work on offloading methods under various environments.

Most offload methods are heuristics algorithms [5], [6], [10], [18], [20], [23]–[28]. B. Li et al. [20] study offload method when the application belongs to computation-intensive applications. MinHop, MET and METComm heuristics were used to offload tasks when the task dynamically arrives independent. L.Yang et al. [24] formulated the offloading problem into an energy-consuming (EC) minimization problem while meeting other constraints. They used ASO and Pro-ITGO to offload tasks. The ASO algorithm is a lightweight linear programming algorithm with three steps: sub-deadline allocation, topology sorting, and task offloading sub-algorithms. The proposed algorithm derives from the original ITGO (Invasive Tumor Growth Optimization) algorithm. Y. Li et al. [5] gave the scheduling target function according to the delay limitation by the potential game equation, and the MD selects MEC nodes according to the game results to offload tasks. For 5G heterogeneous wireless network, S. Han [29] proposed a offload algorithm with a congestion-aware WiFi environment. He proposed a distributed algorithm target to maximizing network utility. Y. Hao et al. [30] given tasks different priorities for offloading tasks according to the energy consumption and urgency. They [6] also consider how to immigrate VMs when every node has the ability to collect green energy. They used the clustering method to decide the location of the VM. And then they proposed a heuristic algorithm to transfer energy, immigrate VM, and allocate tasks. Li Kuan et al. [18] took account of multiple users, multiple offloading points, and structured tasks for offloading tasks. They used genetic algorithm with a greedy strategy to handle the offload tasks. More work can be found in [15,29–34]. Generally speaking, those methods are giving the scheduling targets and finding some heuristics algorithm to meet their scheduling targets. They always pay attentions to the 1-to-1 (one MD and one mobile servers) or 1-to-many environment (one MD and many mobile servers), and proposed methods to offload tasks. Different from the above methods, we pay attention to a many-to-many environment.

Some offloading methods are two or more methods working together to complete the offloading problem. Y. Cui et al. [27] focus on the offloading problem under multi-user and dynamic environments. They tried to solve channel allocation, and offload the computation tasks with an evolutionary game model. They designed an evolutionary game algorithm based on reinforcement learning to offload tasks. W. Tang et al. [4] firstly modeled geo-distributed mobile edge servers in a peer-to-peer networks. Then, they gave offload method consider deadline, cost and so on. The method aims to improve the offloading efficiency while meeting their deadlines.

Some research also finds the offloading method by reinforcement learning methods [21], [25], [27], [28]. M. Hossain et al. [28] used an optimal binary computational offloading decision method to define the offloading problem and then solve the problem by reinforcement learning. Q. Qi et al. [25] formulated the offloading decision for multiple tasks considering a long-term time. They used the recent deep reinforcement learning to solve the offloading problem. The method scruples the future data dependence and continually online learning to improve its performance. J. Wang et al. [37] focused on the offloading problem of the vehicular user (VU) in MEC-enabled vehicular networks. They optimized the offloading problem by considering the time and the energy consumption for transferring files, and local DVFS attributes of local devices. H. Lu et al. [38] focused the offloading problem in the large-scale heterogeneous mobile edge computing environment. They supposed that there are multiple service nodes and the tasks are independent to each other. They solved the offloading problem by using the LSTM network layer and the candidate network set to offload tasks in ab environment of the MEC. Other methods based on reinforcement learning can be found in [39], [40]. In the Vehicular ad-hoc network (VANET), U Maan et al. [41] tried to offload tasks by the estimation of vehicles' future locations. They used Kalman filter prediction scheme to

estimate the vehicle's next location and deep Q network-based reinforcement learning to select the resources-rich fog node in VANET. Those reinforcement learning based offloading methods always give offload method from the view of task. They neglect the fact that a task is offloaded or not to the remote cloud influences the system load of the cloud and the MD, thus influencing whether the coming task can be offloaded to the remote cloud. We try to consider the influence in the paper.

With more attention having been given to the offloading problem in MCE, researchers give new directions to the scheduling targets, such as cost, security, and so on[1], [2], [14], [19], [20], [24]. Those are not related to our paper. Therefore, we do not try to introduce them here.

3. System framework and the related models

In this section, the system framework and the related models are introduced as follows.

3.1 The framework of offload tasks from a MD to remote servers

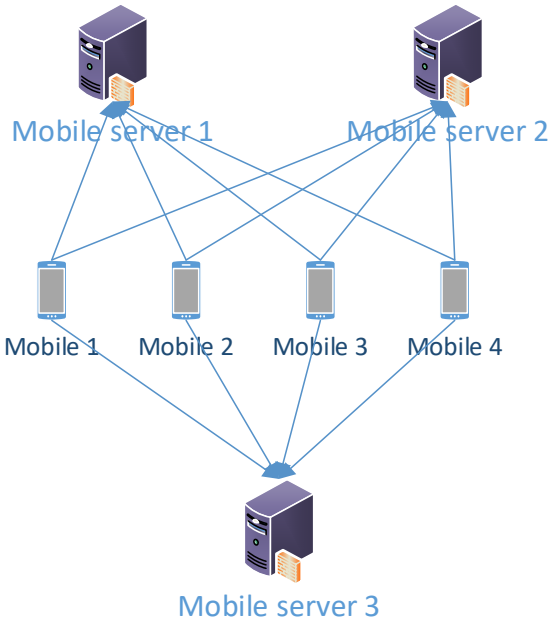

Fig. 1. Multiple MDs and multiple mobile servers

In **Fig. 1**, there are three mobile servers (mobile servers 1~3) and four MDs (MDs 1~4). A MD connects with multiple mobile servers, such as the MD 1 can connect with mobile servers 1~3 with different bandwidths. Different routes between MDs and mobile servers have different metrics in sending/receiving rate, sending/receiving power consumption. So, in scheduling, how to select routes is one of the most important issues to ensure a lower energy consumption for the MD.

The architecture involved in our paper includes four components (**Fig. 2**): tasks, the mobile server manager, the mobile manager, and cloud resources. When the MD gets a task, the mobile scheduling manager will select the execution location according to the offloading method. However, the execution location is based on multiple facts, including the bandwidth between MDs and the remote mobile server, the energy consumption in various places, and other requirements of users. The mobile server manager (MSM) collects the information of

jobs and monitors the system load of the remote mobile server, and also allocates the mobile server for offloading jobs submitted from a MD. The MSM selects a mobile server for every job according to the energy consumption on various remote mobile servers and the system load. It even adds resources dynamically to the system when it is overloaded. With the help of virtualization technology, MSM can add more VMs (as mobile servers) to the system if the system needs.

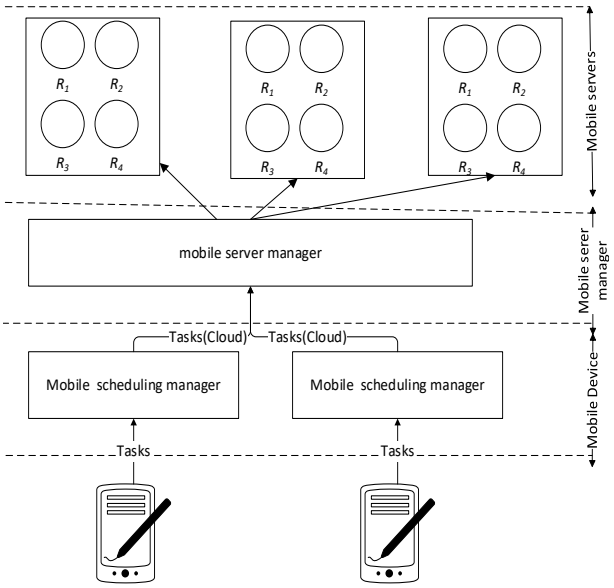

Fig. 2. The framework of offloading tasks

3.2 The related models

Table 1 lists all parameters used in the paper. ND is the number of MDs, NS is the number of mobile servers. For the job J_{nd}^{i} located on ndth MD, DT_{nd}^{i} is the deadline of the job. A_{nd}^{i} is the arriving time. IN_{nd}^{i} and OUT_{nd}^{i} are the size of input files and output files. NI_{nd}^{i} is the number of instructions (million instructions (*MI*)).

$$J_{nd}^{i} = \left\{IN_{nd}^{i}, OUT_{nd}^{i}, A_{nd}^{i}, DT_{nd}^{i}, NI_{nd}^{i}, LC_{nd}^{i}\right\} \tag{1}$$

In formula (1), LC_{nd}^{i} denotes the execution place. If it equals 0, job J_{nd}^{i} is executed on the ndth MD; otherwise, it denotes the selected remote mobile server. CP_{nd} is the computing power consumption of the ndth MD. As to formula (2), it is always denoted by a map between computing voltage V_{nd}^{l}, power consumption E_{nd}^{l} and processing ability P_{nd}^{l}. MP_{nd} is the number of mapping sets.

$$CP_{nd}=<V_{nd}^{l}, E_{nd}^{l}, P_{nd}^{l}> \tag{2}$$

For mobile server S_{ns}, NC_{ns} is the number of cores in nsth mobile server, PA_{ns} is the processing ability of every core in nsth mobile server, L_{ns} is the system load of the mobile server.

$$S_{ns} = \{NC_{ns}, PA_{ns}, LS_{ns}\} \tag{3}$$

The total computational ability of the mobile server is P_{ns}, denoted as follows:

$$P_{ns} = NC_{ns} * PA_{ns} \tag{4}$$

$B(nd, ns)$ is the bandwidth between ndth MD and nsth mobile server.

Table 1. Metrics used in our paper:

Metrics	Meaning
ND	Number of MDs
NS	Number of mobile servers
NT_{nd}	The total number of tasks in ndth MD
J_{nd}^{i}	The ith task from ndth MD
EC_{nd}^{i}	Energy consumption of J_{nd}^{i}
ET_{nd}^{i}	Execution time of J_{nd}^{i}
F_{nd}^{i}	If $F_{nd}^{i} = 1$, it has been finished, otherwise, it isn't.
DT_{nd}^{i}	The deadline of J_{nd}^{i}
A_{nd}^{i}	The arrival time of J_{nd}^{i}
IN_{nd}^{i} and OUT_{nd}^{i}	The size of input files and out files of J_{nd}^{i}
NI_{nd}^{i}	The number of instructions of J_{nd}^{i}
LC_{nd}^{i}	The execution place of J_{nd}^{i}
l	lth work state ndth MD
V_{nd}^{l}	computing voltage of lth work state ndth MD
E_{nd}^{l}	power consumption of lth work state ndth MD
P_{nd}^{l}	processing ability of lth work state ndth MD
CP_{nd}	A mapping set between V_{nd}^{l}, E_{nd}^{l} and P_{nd}^{l}
S_{ns}	The nsth mobile server
NC_{ns}	The number of cores in nsth mobile server
PA_{ns}	The processing ability of every core in nsth mobile server
LS_{ns}	The load of nsth mobile server
LD_{nd}	The load of ndth MD
$SR(nd, ns)$	The sending rate between ndth MD and nsth mobile server
$SE(nd, ns)$	The power consumption for sending files from ndth MD to nsth mobile server
$RR(nd, ns)$	The receiving rate between ndth MD and nsth mobile server
$RE(nd, ns)$	The power consumption for receiving files from ndth MD to nsth mobile server

For the job J_{nd}^{i}, let function $checkl(J_{nd}^{i})$ return whether it has been completed; function $eg(J_{nd}^{i})$ is the energy consumption of job J_{nd}^{i}, so,

Maximizing:

$$Tar_1 = \sum checkl(J_{nd}^{i}) \tag{5}$$

$$Tar_2 = \sum (IN_{nd}^{i} + OUT_{nd}^{i}) * checkl(J_{nd}^{i}) \tag{6}$$

$$Tar_3 = \sum NI_{nd}^{i} * checkl(J_{nd}^{i}) \tag{7}$$

Minimizing:

$$Tar_4 = \sum checkl(J_{nd}^{i}) * eg(J_{nd}^{i}) \tag{8}$$

Subject to:

$$LS_{ns} \leq 1 \tag{9}$$

$$LD_{nd} \leq 1 \tag{10}$$

For $eg(J_{nd}^{i})$, the energy consumption on the MD (when it is executed on the MD) and the energy for transferring files (when it is offloaded to a mobile server) are two important factors for offloading tasks. We do not consider the energy consumption for computation on the mobile server. We have four targets in the scheduling: maximizing the number of completed jobs (formula (5)), maximizing the size of input files and output files (formula (6)),

maximizing the completed instructions (formula (7)) and minimizing the total energy consumptions (formula (8)). Formulas (9) and (10) to ensure every MD and every mobile server is not overload in the scheduling.

When J_{nd}^{i} is executed on the MD, the execution time et_{nd}^{i} and the energy consumption ec_{nd}^{i} (MD working on $sell$th state) are:

$$et_{nd}^{i}=\begin{cases}\frac{NI_{nd}^{i}}{P_{ns}} & if\ LC_{nd}^{i}==0;\\ \frac{IN_{nd}^{i}}{SR(nd,ns)}+\frac{OUT_{nd}^{i}}{RR(nd,ns)}+\frac{NI_{nd}^{i}}{P_{nd}^{sell}} & if\ LC_{nd}^{i}\neq 0.\end{cases} \tag{11}$$

$$ec_{nd}^{i}=\begin{cases}\frac{NI_{nd}^{i}}{P_{ns}}*NC_{ns}*P_{nd}^{sell} & if\ LC_{nd}^{i}==0;\\ \frac{IN_{nd}^{i}}{SR(nd,ns)}*SE(nd,ns)+\frac{OUT_{nd}^{i}}{RR(nd,ns)}*RE(nd,ns) & if\ LC_{nd}^{i}\neq 0.\end{cases} \tag{12}$$

In formula (12), suppose that we select the $sell$th working stage of the MD.

4. An offload heuristic to offload tasks from multiple mobile nodes to mobile servers

In this section, first, we give the offload method to the MD, and then we analyze the complexity of the proposed method.

4.1 Scheduling method for the mobile nodes and mobile servers

As mentioned in formula (12), we get energy consumption when a job is executed on the MD (ecd_{nd}^{i}) (Formula (13)) and on the mobile server (ecs_{nd}^{i}) (Formula (14)):

$$ecd_{nd}^{i}=\frac{NI_{nd}^{i}}{P_{ns}}*NC_{ns}*P_{nd}^{sell} \tag{13}$$

$$ecs_{nd}^{i}=\frac{IN_{nd}^{i}}{SR(nd,ns)}*SE(nd,ns)+\frac{OUT_{nd}^{i}}{RR(nd,ns)}*RE(nd,ns) \tag{14}$$

We judge the preferring location by judging ecm_{nd}^{i} and ecd_{nd}^{i} which has a lower value. We also take account of the execution time on the MD (etm_{nd}^{i}) and the mobile server (etd_{nd}^{i}):

$$etd_{nd}^{i}=\frac{NI_{nd}^{i}}{P_{ns}} \tag{15}$$

$$ets_{nd}^{i}=\frac{IN_{nd}^{i}}{SR(nd,ns)}+\frac{OUT_{nd}^{i}}{RR(nd,ns)}+\frac{NI_{nd}^{i}}{P_{nd}^{sell}} \tag{16}$$

From the above analysis, if ecd_{nd}^{i} is more than ecs_{nd}^{i}, the job J_{nd}^{i} prefers to select the remote server, otherwise, the job prefers to execute on the mobile node. The problem is formulas (14) and (15) have different values when the MDs select different mobile servers (different routes have different bandwidth and energy consumption for transferring files). Here we give a parameter $band(nd,ns)$ to illustrate the bandwidth efficiency:

$$band(nd,ns)=SE(nd,ns)/SR(nd,ns)+RE(nd,ns)/RR(nd,ns) \tag{17}$$

And another parameter ecr_{nd}^{i} to illustrate the energy consumption efficiency on the MD and the mobile server:

$$ecr_{nd}^{i}=ecd_{nd}^{i}/\,ecs_{nd}^{i} \tag{18}$$

For every MD, we suppose that every MD connects with the mobile server with the highest bandwidth efficiency ($band(nd,ns)$), the saving energy is:

$$sec_{nd}^{i}=ecd_{nd}^{i}-ecs_{nd}^{i} \tag{19}$$

For every MD (nd), we have such targets:

Maximizing:

$$ta_1^{nd} = \sum_i sec_{nd}^i * JL_{nd}^i \quad (20)$$

$$ta_2^{nd} = \sum_i ecr_{nd}^i * JL_{nd}^i \quad (21)$$

Subject to:

$$LS_{nd} \leq 1 \quad (22)$$

Formulas (20) and (21) are the total energy consumption efficiency and the total energy saving. For the job J_{nd}^i in the MD (nd), JL_{nd}^i denotes the execution location, if it is, JL_{nd}^i equals 1, otherwise, it is 0. So, the problem becomes a 0-1 integer programming problem.

To reduce the complexity of the problem, we first use FCFS policy for the jobs when the system load (LS_{nd}) of the MD is less than α. We rank every job in every MD according to the ascending order of ecr_{nd}^i, and schedule the job as FCFS policy until the system is more than α. When the system load is more than α, we use the 0-1 integer programming to solve the scheduling problem. Algorithm 1 gives the details:

Algorithm 1: Sch-Device()

1: **For** each MD (nd)
2: Rank jobs in every MD according to the descending order of ecr_{nd}^i;
3: **While** $LS_{nd} \leq$ α
4: Schedule the first job and update the system load LS_{nd} of the MD
5: **Endwhile**
6: **EndFor**
7: **While** (all MD is overload) or (all tasks have been completed)
8: **For** MD nd in ND
9: Solve the scheduling problem in formulas (20) to (22) as a 0-1 programming problem;
10: **For** every job in MD nd
11: **If** $JL_{nd}^i == 1$
12: Schedule job J_{nd}^i on the MD;
13: **EndIf**
14: **EndFor**
15: **EndFor**
16: **EndWhile**

In algorithm 1, first of all, we rank jobs in every MD according to the descending order of ecr_{nd}^i(Line 1, in algorithm 1; same in the following paragraph). Then, we select job one by one until the system load is more than α (lines 2~5). Then, we use the 0-1 programming method to schedule jobs (line 9). Our targets include formulas (20) and (21). If JL_{nd}^i equals 1, the job will be executed on the MD (line 12); otherwise, we drop and send it to the waiting list to be processed on mobile servers.

After that, all jobs need to be offloaded to a mobile server and the problem is how to select which mobile server. Here, we take a suffrage policy: we check every job in the scheduling to find the lowest value of ets_{nd}^i (denoted as $ets1_{nd}^i$), and the second-lowest value ets_{nd}^i (denoted as $ets2_{nd}^i$):

$$suf_{nd}^i = ets2_{nd}^i - ets1_{nd}^i \quad (19)$$

We select the job with the maximizing suf_{nd}^i, and allocate the job to the related mobile server. Algorithm 2 gives the detail.

Algorithm 2: Sch-jobs()

1: $massuf = 0$;
2: **While** the job list is not NULL
3: **For** every job J_{nd}^{i} in the job list
4: Get the lowest value of ets_{nd}^{i} (denoted as $ets1_{nd}^{i}$) when it is offloaded to the mobile server ms_1;
5: Get the second lowest value of ets_{nd}^{i} (denoted as $ets2_{nd}^{i}$) when it is offloaded to the mobile server ms_2;
6: $temp = ets2_{nd}^{i} - ets1_{nd}^{i}$;
7: **If** $temp > massuf$
8: $massuf = temp$;
9: $selj = J_{nd}^{i}$;
10: $sels = ms_1$;
11: **EndIf**
12: **EndFor**
13: Allocate $selj$ to $sels$.
14: Update the system load of every mobile server.
15: **EndWhile**

In Algorithm 2, $massuf$ (Line 1, Algorithm 2; same in the following paragraph) is used to record the maximizing Sufferage [42] of Formula (19). Lines 4 and 5 find the lowest ets_{nd}^{i} and the second lowest ets_{nd}^{i}, denoted by $ets1_{nd}^{i}$ and $ets2_{nd}^{i}$. Line 6 gets the difference between $ets1_{nd}^{i}$ and $ets2_{nd}^{i}$. Lines 7~11 find the job with the lowest suf_{nd}^{i}. $selj$ and $sels$ are the selected job and the selected mobile server. Line 13~14 allocate the selected job $selj$ to the selected mobile server $sels$, and then update the system load of the selected mobile server.

4.2 Complexity analysis

Algorithm 1 has two parts: lines 1-6 and lines 7-16. For the first part, the complexity of lines 1-6 is $O(\max(NT_{nd}))$; NT_{nd} is the total number of tasks in ndth MD. For the second part, because the task that can be allocated to a mobile node is a constant (ensuring the system load is less than α), the complexity of line 9 is $O(1)$. So, the complexity of the second part is $O(1 * ND)$. Thus:

$$O(\text{Algorithm 1}) = O(\max(NT_{nd})) + O(ND)$$

For Algorithm 2, line 4 and line 5 have the same complexity: $O(\max(NT_{nd}))$. Line 3 also has a complexity of $O(\max(NT_{nd}))$. So, the complexity of Algorithm 2 is:

$$O(\text{Algorithm 1}) = O(\max(NT_{nd}) * \max(NT_{nd}))$$

In conclusion, the complexity of our algorithm is:

$$O = O(\max(NT_{nd})) + O(ND) + O(\max(NT_{nd}) * \max(NT_{nd}))$$
$$= O(ND) + O(\max(NT_{nd}) * \max(NT_{nd}))$$

5. Simulations and comparisons

5.1 Simulation environment

The parameters in the simulation environment are given in **Table 2**. We suppose that the system has 20000 jobs, and the number of instructions of each job is a random in [1 100000]*MI* (Million instructions). The file size of the input and output files is in the scope of [0 1000]*M*. Each MD has 4 cores and each has a random computing speed in 1800~2200*MHz*. The working power consumption of the MD is random in 0.4~0.6*W* and the idle power consumption is 0.001W. Each mobile server has 16 cores and each has a random computing

speed in 2000~3000 *MHz*. The sending rate between the MD and mobile server is a random number in 6~8*M*/*s* and the receiving rate is a random number in 10~20 *M*/*s*. The average power consumption for receiving data and average power consumption for sending data is 0.05W and 0.1W. The deadline of jobs is 1.4~1.8 (Random) times of the execution time. All the results are the average value of 100 times. The average arrival rate (AAR) (per. hour) is changed from 400 to 500 with a step of 10. There are 10 MDs and 10 mobile servers in the simulation environment. We will evaluate five metrics of the four methods: AET (Average execution time), NCJ (Number of completed jobs), AEC (Average energy consumption), NOI (number of instructions of completed tasks) and FS (files size of completed jobs). According to the parameters used in **Table 1**, the five metrics are given as follows:

$$NCJ = \sum_i \sum_{nd} F_{nd}^i$$
$$AET = \sum_i \sum_{nd} ET_{nd}^i / NCJ$$
$$AEC = \sum_i \sum_{nd} EC_{nd}^i / NCJ$$
$$NOI = \sum_i \sum_{nd} F_{nd}^i * NI_{nd}^i / NCJ$$
$$FS = \sum_i \sum_{nd} F_{nd}^i * (OUT_{nd}^i + IN_{nd}^i) / NCJ$$

Table 2. Parameters used in the simulation

Parameters	Values
Working frequency of the MD (4 cores)	1800~2200*MHz*
Working frequency of mobile server (16 cores)	2000~3000*MHz*
Working power consumption of the MD	0.4~0.6*W*
Idle power consumption of the MD	0.001*W*
Average power consumption for receiving data (100M/s)	0.05*W*
Average power consumption for sending data (10M/s)	0.1*W*
Receiving data rate (from MD to mobile server)	50~150M/s
Seeding data rate (from MD to mobile server)	20~80M/s

5.2 Comparisons and discussions

We will compare our method with Tradeoff (TDO) [43]–[45], adaptive offloading (AO) [16] and Dynamic Programming-based Energy Saving Offloading (DPESO) algorithm [46]. DPESO gave the offloading method by considering the offloading option, offloading sequence and transmission power by judging the location where the job saves more energy. The TDO is an intelligent computation offloading system that makes tradeoff decisions for code offloading from the MD to the cloud to reduce energy consumption. AOD uses a fitness function to evaluate the offloading scheme. We will compare those four methods in the following metrics: number of un-completed jobs (NUJ) (**Fig. 3**), average energy consumption (AEC) (**Fig. 4**), average execution time (AET) (**Fig. 5**), number of instructions of completed tasks (NOI) (**Fig. 6**) and files size of completed tasks (FS) (**Fig. 7**). We call our method as "Sufferage" in the simulation which includes Algorithm 1 and Algorithm 2.

Fig. 3 is the AET of DPESP, AO, and TDO. Sufferage always has the lowest value in AET under any AARs. The AET of Sufferage, DPESP, AO, and TDO is 6.3528 (*s*), 6.8672 (*s*), 7.4151 (*s*), and 7.6302 (*s*), respectively. **Fig. 4** is the AEC of those methods when AARs are changed from 400 to 500 with a step of 10. The ascending order of AECs of those four methods is Sufferage, DPESP, AO, and TDO. To AECs of DPESP, AO, and TDO, Sufferage average reduces 0.3314, 0.2479, and 0.6191, about 10.39%, 8.43%, and 24.09%. Sufferage performs best in AEC and AET because the scheduling algorithm in the MD ensures the lowest cost and the scheduling algorithm in the mobile server considers the future influence on the scheduling order. When Sufferage offloads tasks, it makes that the task has a lower execution time and

energy consumption when the task is executed on the MD. Moreover, when the task is offloaded to the mobile server, it considers the influence of scheduling tasks. Other methods just consider the scheduling result and give a static judgment (such as tradeoff, fit function, and others) for the scheduling. Therefore, sometime, the scheduling of some tasks makes the coming tasks do not have a lower energy consumption in the future.

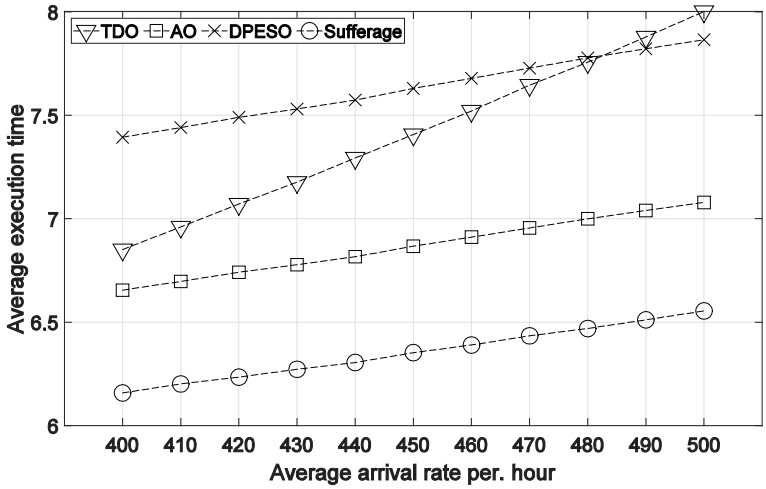

Fig. 3. AET under different AARs

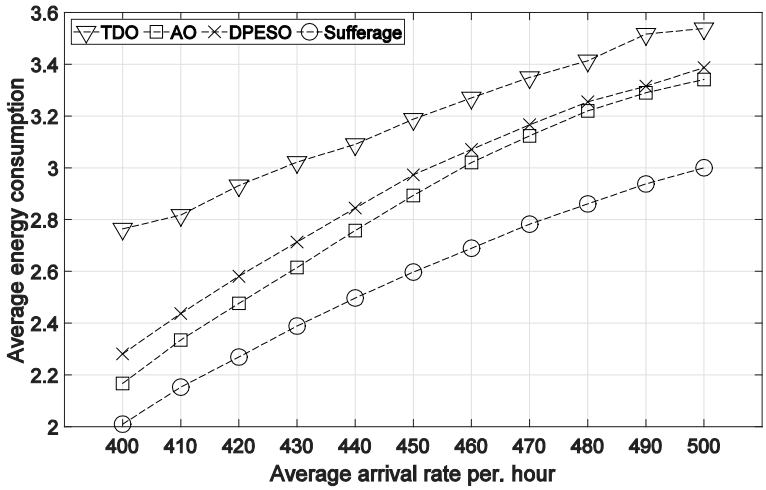

Fig. 4. AEC under different AEC

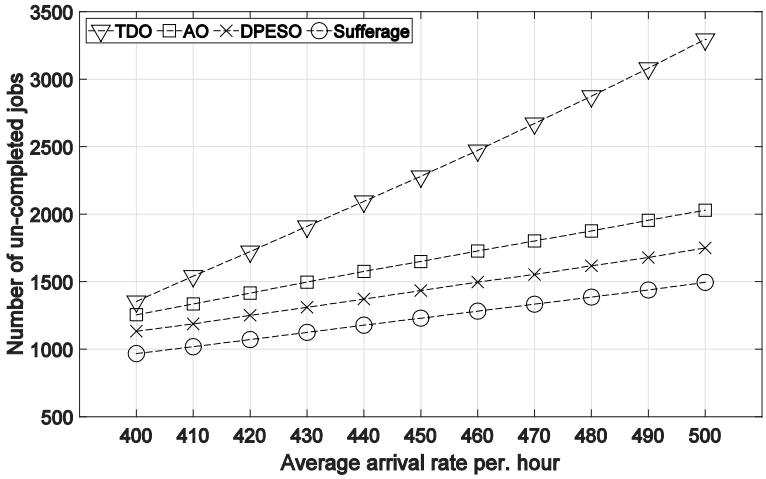

Fig. 5. NUJ under different AARs

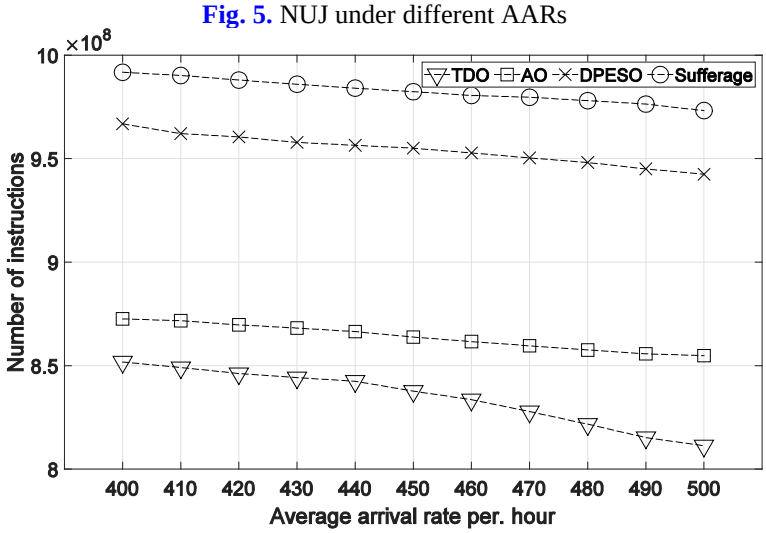

Fig. 6. NOI under different AAR

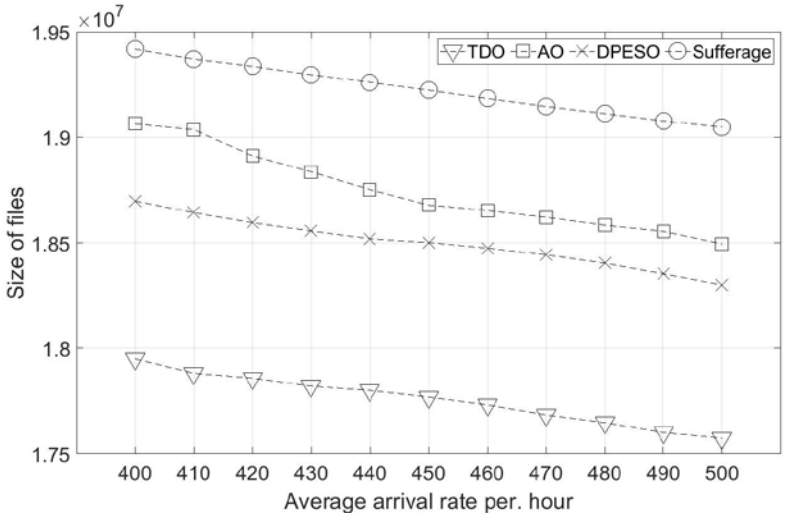

Fig. 7. SF under different AAR

Fig. 5 is the NUJ of those four methods under different AARs. Sufferage always has the lowest value in NUJs, followed by DPESP, AO, and TDO. To the NUJ of DPESP, AO and TDO, Sufferage average reduces 206.6804, 413.2690, and 1072.8, about 14.40%, 25.17% and 46.61%. **Fig. 6** and **Fig. 7** are the NOI and FS of all methods. In general, the four methods in **Fig. 6** and **Fig. 7** have the same trend with NUJ in **Fig. 6.** Compared to NOI of DPESP, AO and TDO, Sufferage average reduces (in billion MI) 2.4794, 2.5434 and 0.2954, respectively. Compared to SF of DPESP (**Fig. 7**), AO and TDO, Sufferage average reduces (in ten thousand *M*) by 7.5965, 4.1228e and 1.5004, respectively. Generally speaking, all those methods have an increasing trend in NUJ (**Fig. 3**), AET (**Fig. 5**) and AEC (**Fig. 4**), and have a decreasing trend in NOI (**Fig. 6**) and FS (**Fig. 7**) with the enhancement of AARs. With the enhancement of AARs, the system has a higher system load, more tasks cannot be completed before their deadlines, thus enhancing NUJ and decreasing FS and NOI. It also makes the system have a larger value in AET and ACE. Sufferage makes full of use local and the mobile server resources, thus having a minimum value in NUJ. At the same time, when the total number of tasks is a constant, Sufferage may have the highest value in NI and FS.

6. Conclusion and future work

In this paper, we focus on the offloading problem when there are multiple MDs and multiple mobile servers. Based on the analysis, first, we select the tasks executed on the MD according to the ration between the energy consumption on the MD and the remote cloud; for the task executed on the cloud, we propose a suffrage heuristic to offload tasks and select the mobile server. Simulation results show that our proposed method improves the number of completed jobs, and reduce energy consumption. We know that the mobile node may have the ability to harvest energy, so how to use the energy and reduce the energy consumption from other system is the two key problems. We may pay some attention to the new environment and give offloading method. As to the future work, we may use Image Retagging technology [47], [48] to manage the network of the MD for task offloading.

Reference

[1] H. Lin, S. Zeadally, Z. Chen, H. Labiod, and L. Wang, “A survey on computation offloading modeling for edge computing,” *J. Netw. Comput. Appl.*, vol. 169, no. July, p. 102781, 2020. Article (CrossRef Link)

[2] K. Kumar, J. Liu, Y. H. Lu, and B. Bhargava, “A survey of computation offloading for mobile systems,” *Mob. Networks Appl.*, vol. 18, no. 1, pp. 129–140, 2013. Article (CrossRef Link)

[3] J. Lu, Y. Hao, K. Wu, Y. Chen, and Q. Wang, “Dynamic offloading for energy-aware scheduling in a mobile cloud,” *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3167–3177, 2022, Article (CrossRef Link)

[4] W. Tang, X. Zhao, W. Rafique, L. Qi, W. Dou, and Q. Ni, “An offloading method using decentralized P2P-enabled mobile edge servers in edge computing,” *J. Syst. Archit.*, vol. 94, pp. 1–13, 2019. Article (CrossRef Link)

[5] Y. Li and C. Jiang, “Distributed task offloading strategy to low load base stations in mobile edge computing environment,” *Comput. Commun.*, vol. 164, pp. 240–248, 2020. Article (CrossRef Link)

[6] Y. Hao, J. Cao, Q. Wang, and J. Du, “Energy-aware scheduling in edge computing with a clustering method,” *Futur. Gener. Comput. Syst.*, vol. 117, pp. 259–272, 2021. Article (CrossRef Link)

[7] B. B. Bista, J. Wang, and T. Takata, "Probabilistic computation offloading for mobile edge computing in dynamic network environment," *Internet of Things*, vol. 11, p. 100225, 2020. Article (CrossRef Link)
[8] W. Huang, K. Ota, M. Dong, T. Wang, S. Zhang, and J. Zhang, "Result return aware offloading scheme in vehicular edge networks for IoT," *Comput. Commun.*, vol. 164, pp. 201–214, 2020. Article (CrossRef Link)
[9] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," *IEEE Commun. Surv. Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017. Article (CrossRef Link)
[10] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber-wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, 2018. Article (CrossRef Link)
[11] Y. Hao, Q. Wang, J. Cao, T. Ma, J. Du, and X. Zhang, "Interval grey number of energy consumption helps task offloading in the mobile environment," *ICT Express*, 2022. Article (CrossRef Link).
[12] K. Li, "Computation Offloading Strategy Optimization with Multiple Heterogeneous Servers in Mobile Edge Computing," *IEEE Trans. Sustain. Comput.*, pp. 1–1, 2019. Article (CrossRef Link)
[13] M. Wang, L. Zhu, L. T. Yang, M. Lin, X. Deng, and L. Yi, "Offloading-assisted energy-balanced IoT edge node relocation for confident information coverage," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4482–4490, 2019. Article (CrossRef Link)
[14] X. Chen, S. Chen, Y. Ma, B. Liu, Y. Zhang, and G. Huang, "An adaptive offloading framework for Android applications in mobile edge computing," *Sci. China Inf. Sci.*, vol. 62, no. 8, pp. 1–17, 2019. Article (CrossRef Link)
[15] E. El Haber, T. M. Nguyen, and C. Assi, "Joint Optimization of Computational Cost and Devices Energy for Task Offloading in Multi-Tier Edge-Clouds," *IEEE Trans. Commun.*, vol. 67, no. 5, pp. 3407–3421, 2019. Article (CrossRef Link)
[16] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities," *Futur. Gener. Comput. Syst.*, vol. 105, pp. 717–729, 2020. Article (CrossRef Link)
[17] L. Chen, X. Li, H. Ji, and V. C. M. Leung, "Computation offloading balance in small cell networks with mobile edge computing," *Wirel. Networks*, vol. 25, no. 7, pp. 4133–4145, 2019. Article (CrossRef Link)
[18] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *J. Supercomput.*, vol. 71, no. 8, pp. 3009–3036, 2015. Article (CrossRef Link)
[19] J. Long, Y. Luo, X. Zhu, E. Luo, and M. Huang, "Computation offloading through mobile vehicles in IoT-edge-cloud network," *Eurasip J. Wirel. Commun. Netw.*, vol. 2020, no. 1, 2020. Article (CrossRef Link)
[20] X. Wei et al., "MVR: An Architecture for Computation Offloading in Mobile Edge Computing," in *Proc. of 2017 IEEE 1st Int. Conf. Edge Comput. EDGE 2017*, pp. 232–235, 2017. Article (CrossRef Link)
[21] L. Yang, C. Zhong, Q. Yang, W. Zou, and A. Fathalla, "Task offloading for directed acyclic graph applications based on edge computing in Industrial Internet," *Inf. Sci. (Ny).*, vol. 540, pp. 51–68, 2020. Article (CrossRef Link)
[22] Q. Qi et al., "Knowledge-Driven Service Offloading Decision for Vehicular Edge Computing: A Deep Reinforcement Learning Approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 5, pp. 4192–4203, 2019. Article (CrossRef Link)
[23] S. K. Dash, S. Dash, J. Mishra, and S. Mishra, "Opportunistic Mobile Data Offloading Using Machine Learning Approach," *Wirel. Pers. Commun.*, vol. 110, no. 1, pp. 125–139, 2020. Article (CrossRef Link)
[24] Y. Cui, D. Zhang, T. Zhang, L. Chen, M. Piao, and H. Zhu, "Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices," *AEU - Int. J. Electron. Commun.*, vol. 118, p. 153134, 2020. Article (CrossRef Link)

[25] M. S. Hossain, C. I. Nwakanma, J. M. Lee, and D. S. Kim, "Edge computational task offloading scheme using reinforcement learning for IIoT scenario," *ICT Express*, vol. 6, no. 4, pp. 291–299, 2020. Article (CrossRef Link)
[26] F. Xu, W. Yang, and H. Li, "Computation offloading algorithm for cloud robot based on improved game theory," *Comput. Electr. Eng.*, vol. 87, pp. 1–11, 2020. Article (CrossRef Link)
[27] S. Han, "Congestion-aware WiFi offload algorithm for 5G heterogeneous wireless networks," *Comput. Commun.*, vol. 164, pp. 69–76, 2020. Article (CrossRef Link)
[28] Y. Hao, J. Cao, Q. Wang, and T. Ma, "Energy-aware offloading based on priority in mobile cloud computing," *Sustain. Comput. Informatics Syst.*, vol. 31, p. 100563, 2021. Article (CrossRef Link)
[29] A. Hekmati, P. Teymoori, T. D. Todd, D. Zhao, and G. Karakostas, "Optimal multi-part mobile computation offloading with hard deadline constraints," *Comput. Commun.*, vol. 160, pp. 614–622, 2020. Article (CrossRef Link)
[30] Y. Hao, J. Cao, Q. Wang, and T. Ma, "Energy-aware offloading based on priority in mobile cloud computing," *Sustain. Comput. Informatics Syst.*, vol. 31, p. 100563, 2021. Article (CrossRef Link)
[31] X. Chen et al., "Cooling-Aware Optimization of Edge Server Configuration and Edge Computation Offloading for Wirelessly Powered Devices," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5043–5056, 2021. Article (CrossRef Link)
[32] W. Zhou, L. Xing, J. Xia, L. Fan, and A. Nallanathan, "Dynamic Computation Offloading for MIMO Mobile Edge Computing Systems with Energy Harvesting," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 5172–5177, 2021. Article (CrossRef Link)
[33] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, and X. Chen, "Dynamic Offloading and Resource Scheduling for Mobile Edge Computing With Energy Harvesting Devices," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 2154–2165, 2021. Article (CrossRef Link)
[34] A. Asheralieva and T. D. Niyato, "Fast and Secure Computational Offloading with Lagrange Coded Mobile Edge Computing," *IEEE Trans. Veh. Technol.*, vol. 70, no. 5, pp. 4924–4942, 2021. Article (CrossRef Link)
[35] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading Tasks with Dependency and Service Caching in Mobile Edge Computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2777–2792, 2021. Article (CrossRef Link)
[36] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. S. Quek, "Computation Offloading for Mobile Edge Computing Enabled Vehicular Networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019. Article (CrossRef Link)
[37] H. Lu, C. Gu, F. Luo, W. Ding, and X. Liu, "Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning," *Futur. Gener. Comput. Syst.*, vol. 102, pp. 847–861, 2020. Article (CrossRef Link)
[38] X. Zhao, Q. Zong, B. Tian, B. Zhang, and M. You, "Fast task allocation for heterogeneous unmanned aerial vehicles through reinforcement learning," *Aerosp. Sci. Technol.*, vol. 92, pp. 588–594, 2019. Article (CrossRef Link)
[39] R. Zhao, X. Wang, J. Xia, and L. Fan, "Deep reinforcement learning based mobile edge computing for intelligent Internet of Things," *Phys. Commun.*, vol. 43, p. 101184, 2020. Article (CrossRef Link)
[40] U. Maan and Y. Chaba, "Deep Q-Network based fog Node Offloading strategy for 5G Vehicular Adhoc Network," *Ad Hoc Networks*, vol. 120, p. 102565, 2021. Article (CrossRef Link)
[41] E. K. Tabak, B. B. Cambazoglu, and C. Aykanat, "Improving the performance of independenttask assignment heuristics minmin,maxmin and sufferage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1244–1256, 2014. Article (CrossRef Link)
[42] M. E. Khoda, M. A. Razzaque, A. Almogren, M. M. Hassan, A. Alamri, and A. Alelaiwi, "Efficient Computation Offloading Decision in Mobile Cloud Computing over 5G Network," *Mob. Networks Appl.*, vol. 21, no. 5, pp. 777–792, 2016. Article (CrossRef Link)
[44] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-Efficient UAV-Assisted Mobile Edge Computing: Resource Allocation and Trajectory Optimization," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 3, pp. 3424–3438, 2020. Article (CrossRef Link)

[45] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proc. of IEEE INFOCOM*, pp. 190–194, 2013. Article (CrossRef Link)
[46] Y. Zhang and J. Fu, "Energy-efficient computation offloading strategy with tasks scheduling in edge computing," *Wirel. Networks*, vol. 27, no. 1, pp. 609–620, 2021. Article (CrossRef Link)
[47] J. Tang et al., "Tri-Clustered Tensor Completion for Social-Aware Image Tag Refinement," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 8, pp. 1662–1674, 2017. Article (CrossRef Link)
[48] J. Tang, X. Shu, Z. Li, Y. G. Jiang, and Q. Tian, "Social Anchor-Unit Graph Regularized Tensor Completion for Large-Scale Image Retagging," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 2027–2034, 2019. Article (CrossRef Link)

Tao Zhang is sensor engineer of Changde City Tobacco Company. His work focuses on CISCO, CCNP and other kinds of network system. His research interesting includes the resource scheduling in Cloud computing, edge computing and other system.

Mingfeng Cao He Mainly engaged in tobacco production and cultivation aspects of research. He has published some papers about the information system about the management of tobacco product.

Yongsheng Hao received his MS Degree of Engineering from Qingdao University in 2008. Now, he is a sensor engineer of Information management department, Nanjing University of Information Science & Technology. His current research interests include distributed and parallel computing, mobile computing, Grid computing, web Service, particle swarm optimization algorithm and genetic algorithm. He has published 50 papers in international conferences and journals.