Churn-Resilient Task Scheduling in a Tiered IoT Infrastructure

Jianhua Fan[1], Xianglin Wei[1,*], Tongxiang Wang[1], Tian Lan[2], Suresh Subramaniam[2]

[1] Nanjing Telecommunication Technology Research Institute, Nanjing, 210007, China
[2] George Washington University, Washington DC, 20052, US
* The corresponding author, email: wei_xianglin@163.com

Abstract: Cloud-as-the-center computing paradigms face multiple challenges in the 5G and Internet of Things scenarios, where the service requests are usually initiated by the end-user devices located at network edge and have rigid time constraints. Therefore, Fog computing, or mobile edge computing, is introduced as a promising solution to the service provision in the tiered IoT infrastructure to compensate the shortage of traditional cloud-only architecture. In this cloud-to-things continuum, several cloudlet or mobile edge server entities are placed at the access network to handle the task offloading and processing problems at the network edge. This raises the resource scheduling problem in this tiered system, which is vital for the promotion of the system efficiency. Therefore, in this paper, a scheduling mechanism for the cloudlets or fog nodes are presented, which takes the mobile tasks' deadline and resources requirements at the same time while promoting the overall profit of the system. First, the problem at the cloudlet, to which IoT devices offload their tasks, is formulated as a multi-dimensional 0-1 knapsack problem. Second, based on ant colony optimization, a scheduling algorithm is presented which treat this problem as a subset selection problem. Third, to promote the performance of the system in the dynamic environments, a churn-refined algorithm is further put forward. A series of simulation experiments have shown that out proposal outperforms many state-of-the-art algorithms in both profit and guarantee ratio.

Keywords: fog computing; task scheduling; deadline constrained; internet of things; ant colony optimization

I. INTRODUCTION

In the 5G and Internet of Things (IoT) era, where the time requirements of the mobile tasks initiated at the network edge are usually rigid, traditional cloud-as-the-center computing paradigm faces several challenges to meet the time-sensitive tasks [1]. For instance, many virtual reality (VR)-based tasks require sub-millisecond response time.

In this backdrop, many edge-centered novel computing paradigms are put forward, including fog computing, mobile edge computing, sensor cloud, and etc. In the fog computing architecture, many fog nodes, or named cloudlet entities, are implemented at the access networks of the wireless IoT devices. They usually have much more powerful computational and communication capabilities than ordinary mobile devices, and can handle the local computational needs on behalf of

Received: Oct. 8, 2018
Revised: Jan. 21, 2019
Editor: Yulong Zou

cloud-infrastructures. Furthermore, they can transfer the received tasks to the cloud entity if the time requirements of the tasks can be fulfilled. Compared with traditional two-layer infrastructure, this tiered system can better fulfill end-user needs in four aspects. First, it can achieve reduced service latency compared with always offloading to remote cloud through the Internet. Second, the coverage area of this architecture will be larger and much more easy to access. Third, it can support the mobility of mobile devices. Fourth, better heterogeneity can be included or tolerated in this architecture [2]. Figure 1 shows an example of this three-layer architecture. Many IoT devices are interconnected by wireless networks, which are constructed based on IEEE 802.11, 802.15, and other standards, and a cloudlet entity locates at the access point of each access network. On the top level, cloudlets connect to the centralized cloud data center through the Internet.

To promote the performance of the tasks requires the system to efficiently utilize the resources at the cloudlet entity, the cloud, as well as the idle resources at mobile IoT devices. Note that these three types of computation facilities have different characteristics [3]. The resources at the mobile devices are usually very limited, but they are much closer to the task initiators. The cloudlet entity has much more resources to schedule, but is also limited compared with that of the cloud due to cost and volume consideration. The cloud has nearly unlimited resources, but offloading to the cloud usually incur long transmission delay. Therefore, a comprehensive scheduling of the available resources is in great need in this architecture. The unique dynamic nature introduced by the wireless network and IoT devices even make this problem more challenging than traditional cloud paradigm which is usually static and has refined fault management mechanism.

The overall objective of this paper is to design a scheduling algorithm for the cloudlet entities to help them utilize all the available resources while meeting mobile tasks' rigid time requirements. The profit of the fog system is the optimization objective, and available resources and reliability of the service providers are taken as the constraints. Furthermore, to cope with the challenges brought by dynamic joining and leaving of service providers, i.e. churn in the system, the failure probabilities of service providers will be considered when designing the heuristic in the scheduling algorithm based on Ant Colony Optimization (ACO). To summary, a multi-dimensional 0-1 knapsack problem is built, and an ACO-based scheduling algorithm is put forward. To cope with the dynamics of end users' IoT devices, a refined algorithm is designed to promote the scheduling performance when IoT devices join and leave the system dynamically. Compared with the algorithm based on ACO [25], the algorithm presented in this paper can better cope with the churn and dynamic nature of the wireless network, which is not considered before. To validate the performance of the algorithm, a series of simulations are conducted on the developed simulator. Remarkable promotion can be observed from the simulation results through compared with state-of-the-art algorithms.

The contributions of this manuscript mainly include:

1) The scheduling problem in the cloudlet entity is formulated as a 0-1 knapsack problem.

2) An efficient scheduling algorithm based on ACO is proposed, which can maximize the

To effectively utilize the available resources, this paper formulates the task scheduling problem as a multi-dimensional 0-1 knapsack problem, and put forwards an algorithm based on Ant Colony Optimization (ACO).

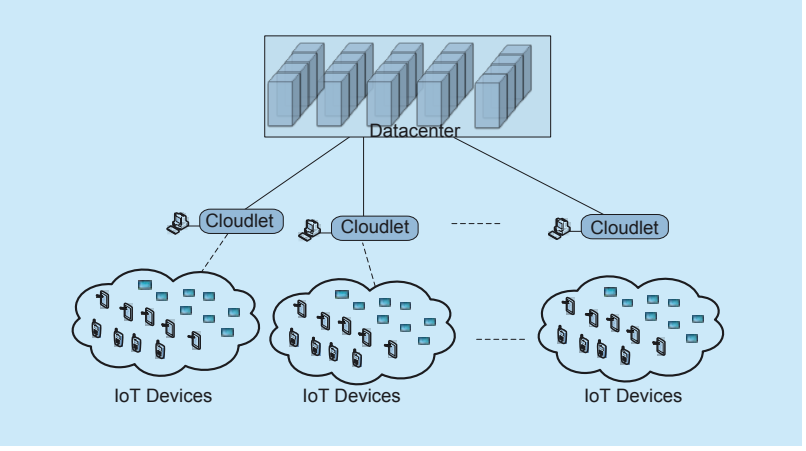


Fig. 1. *Three layer structure of a tiered IoT architecture.*

system profit while meeting the tasks' deadline constraints.

3) To tackle the dynamics nature of the wireless IoT environment, a churn-resilient algorithm is presented which can promote the performance when fog nodes can join and leave dynamically.

4) Extensive simulation experiments are conducted to validate the effectiveness of this paper's proposal.

II. Related Work

As far as we know, the studies on the scheduling problem in Cloud computing are quite abundant and a series of scheduling strategies have been put forward while taking different types of tasks into consideration [3]. However, these achievements cannot be utilized in the scenario of Fog computing directly due to the distributed heterogeneity of the Fog computing model and limited resources of fog nodes [4].

In order to deal with the scheduling problem in Fog-based computing systems, a few scheduling algorithms have been presented by researchers from different perspectives. Pham et al. paid attention to the cloud-fog computing system and put forward a heuristic algorithm [5]. The collaborative execution between cloud data center and fog nodes was investigated by balancing the makespan and monetary cost of the system. Pu et al. presented a novel offloading framework via network-assisted Device-to-Device collaboration [6], whose objective is to minimize the energy consumption. Through the assistance of network operator, mobile users can collaborate with each other to execute tasks by sharing the common resources. The Fog-computing supported embedded system was investigated by Zeng et al., and a joint optimization method was presented to deal with the problems of task scheduling and image placement [7]. However, the above researches did not consider the deadline of task, which may impact the Quality of Service (QoS) seriously.

Existing scheduling algorithms, which take the deadline of each task into consideration, usually aim at Cloud-based systems [8-16]. For example, the genetic algorithm (GA) was employed by Chen et al. to minimize the execution cost when considering the deadline-constrained workflow [8]. In addition, they investigated the Ant Colony System (ACS) based scheduling approach to optimize the execution cost while satisfying the deadline of each task [9]. Zuo et al. also employed the ACS-based method to optimize the resource utilization considering both the deadline and cost of tasks [16].

As for the utilization of Fog or Cloud computing in the IoT system, many researchers contribute a lot. Sarkar et al. conducted the employment of Fog-computing and Cloud-computing in the tiered IoT system [17]. Experimental results showed that as for the IoT system with large numbers of latency-sensitive tasks, Fog-computing was more efficient than Cloud-computing. A resource management framework was presented by Aazam et al. based on the characteristics of customers in IoT system [18]. The industrial IoT was studied by Tang et al. and a GA-based algorithm was proposed to solve the scheduling problem [19]. Yang et al. studied the homogeneity fog system and presented a scheduling method to improve the energy efficient [20].

ACS-based method is an efficient approach to solve the problem of tasks scheduling in Fog-computing system due to its distributed character. Dorigo et al. firstly investigated the application of the ACS in the Traveling Salesman Problem (TSP) [21], which proved that the ACS outperformed the other nature-inspired methods. Wei et al. modeled the tasks scheduling problem to the TSP based on the proposed Hybrid Local Mobile Cloud Model (HLMCM) and then, the ACS was utilized to solve it in [22] and [23]. In order to evaluate the performance of the proposed algorithm, a series of simulations were conducted and the results demonstrated the better performance of the proposed algorithms. However, the deadline was not considered in the scheduling process. Later, they took further studies on the scheduling problem and extended the Fog-

based system to the Cloud-fog system. The performance of ACS-based algorithm was improved and the deadline of task was taken into consideration [25]. However, the dynamic characteristics of Fog nodes were not considered in the scheduling process.

III. Problem Formulation

3.1 Task offloading

As shown in figure 1, a large number of IoT devices may connect to the same cloudlet. Many tasks could be offloaded to a cloudlet by its served IoT devices with different resources requirements. Upon receiving the offloaded tasks from IoT devices, the cloudlet is in charge of scheduling them based on available resources locally and remotely. Moreover, a cloudlet will make a scheduling decision every once in a while, called a timeslot. For any task, it will experience one of the five choices below at a given timeslot:

- Local execution at the cloudlet;
- Further offload it to the remote datacenter;
- Deliver it to an IoT service provider;
- Store it in the task queue for later scheduling ;
- Reject it due to resource limitation.

The scheduling algorithm at the cloudlet has to decide whether and how to execute newly arrived and buffered tasks. It has to select the most suitable execution place for each task based on resources requirements and deadline limitations while maximizing the profit of the system. Figure 2 shows the workflow of the scheduling problem. We can see that tasks arrived at a time interval will be scheduled together with those tasks stored in the buffer. To ensure a continuous task scheduling, the algorithm will be executed at the end of each time interval.

3.2 Assumptions and notations

Before diving into problem statement and algorithm design, we make a few assumptions about the service providers in the system to facilitate problem formulation. First, the remote data center locates in the core network, and has nearly unlimited resources. In comparison, the distance between a use-end device and a cloudlet is usually only one hop. However, the computation, communication, and storage resources at the cloudlet are much less than those at the data center. Second, several IoT devices with idle resources can also be service providers in the system. They can join or leave the system based on their remaining battery supply, wireless link condition or mobility requirements. This assumption is due to the mobility nature of IoT devices and the wireless network's inherent randomness characteristics. Moreover, a server may stop providing service if its own resources are insufficient. Multiple tasks could be executed by some particular service provider at each time interval as long as the total resources requirements of these tasks do not exceed the provider's service capacity. For example, the provider can implement this through initializing different VMs for different tasks. For simplicity of expression, all the service providers will be called hosts in the following analysis. Third, all the submitted tasks are assumed to have known resource requirements, which can be estimated based on execution history or application profiling. Fourth, the profit brought by each task is dependent on its execution time and deadline requirement. No profit can be harvested if it is rejected or not finished before its deadline. Finally, the running time is divided into several time slots, and the scheduling algorithm will

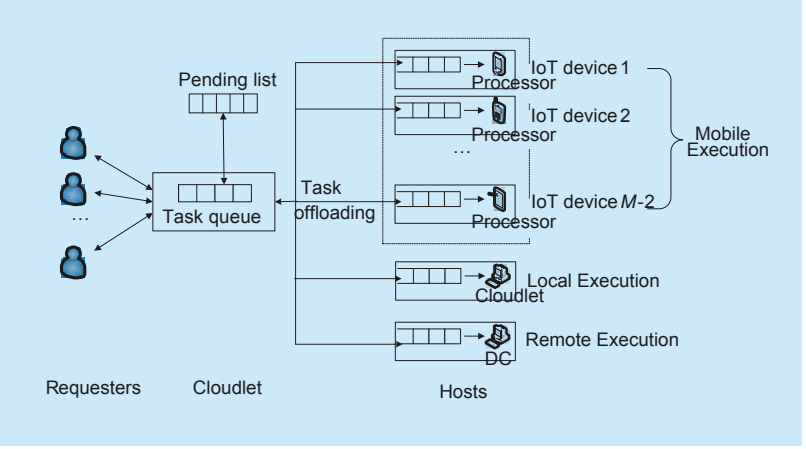


Fig. 2. *The scheduling workflow at the cloudlet entity.*

be executed at each time slot. Table I shows the symbols used for the problem statement.

Table I. *Notations.*

Notation	Meaning
T_p	Buffered tasks set
T_h	Set of tasks executed on host h
h_{user}	All user-end devices
$h_{cloudlet}$	Cloudlet entities
h_{cloud}	Service providers in the data center
H	The union of all hosts, $H=\{h_{user}\}\cup\{h_{cloudlet}\}\cup\{h_{cloud}\}$
$R(h)$	Resources at host h, including computation, communication, and storage, $R(h)=(R_c(h),R_b(h),R_s(h))$
t	A task
M-2	Total number of user-end devices.The total number of hosts is M, including user-end devices, a cloudlet, and a cloud data center.
$r(t)$	Task t's resource requirements, $r(t)=(r_c(t),r_b(t),r_s(t))$
$DS(t)$	Task t's dataset size
T_0	Time for executing a data unit through one unit of computing resource.
B_C	The bandwidth between an IoT device and the cloudlet
B_D	The bandwidth between the datacenter and the cloudlet
T_D	The delay introduced by the transmission path between the datacenter and the cloudlet
T_C	The delay introduced by the transmission path between an IoT device and the cloudlet
T'_D	The delay jitter introduced by the transmission path between the datacenter and the cloudlet
T'_C	The delay jitter introduced by the transmission path between an IoT device and the cloudlet
$x(t,h)$	Task t is scheduled on host h or not, and it can be 0 or 1
T_S	Scheduling time
$a(t)$	t's arrival time
$d(t)$	t's deadline
$l(t)$	t's estimated execution time
$e(t, h)$	t's estimated ending time on host h
$m(t,h)$	t's estimated transmission time from and to host h
α_d	The reward factor of device d which offloads a task to the fog system
$p(t,h)$	The profit of running task t on host h
$b(h)$	The benefit of consuming one unit resource while running a task on host h
$c(h)$	The cost of consuming one unit resource while running a task on host h
$q(h)$	Link quality of host h
$e(h)$	Remaining energy supply of host h
$L(h)$	Leaving factor of host h
$l(h)$	The leaving probability of host h
$h(l)$	The host on link l

3.3 Profits and execution time estimation

Through providing services to IoT devices, the service hosts could obtain some kind of profit. Task t's profit when it is executed on host h is defined as:

$$p(t,h)=(b(h)-c(h))\times r(t)\times(1+\alpha_d\frac{d_t-e(t,h)}{d_t-a_t}) \quad (1)$$

where b(h) and c(h) are the basic benefit and cost of running a task on host h, and α_d is a reward factor that can be used to encourage the system to execute the task, since the smaller $e(t,h)$ is, the higher $p(t,h)$ will be. $e(t,h)$ can be calculated according to (2) based on three parts, i.e. t's start time a_t, execution time l(t), and transmission time $m(t,h)$. l(t) and $m(t,h)$ in (2) are shown in (3) and (4) respectively.

$$e(t,h)=T_S+l(t)+m(t,h) \quad (2)$$

$$l(t)=\frac{T_0\times DS(t)}{r_c(t)} \quad (3)$$

where $DS(t)$ is t's dataset size transmitted in the offloading process.

$$m_t(t,h)=\begin{cases}\frac{DS(t)}{B_C}+\frac{DS(t)}{B_D}+2T_D+2T_C+2T'_D+2T'_C,\\ \text{if } h \text{ is the datacenter}\\ \frac{DS(t)}{B_C}+2T_C+2T'_C,\\ \text{if } h \text{ is the Cloudlet}\\ 2\frac{DS(t)}{B_C}+4T_C+4T'_C,\\ \text{if } h \text{ is a IoT device}\end{cases} \quad (4)$$

Here, T_D is determined by the transmission link between the cloudlet and the data center, and is usually much larger than T_C.

3.4 Problem statement

Our optimization objective in this paper is maximizing the profits while meeting the deadline limitation, and the problem can be formally stated as:

$$\text{Maximize } V=\sum_{t\in Tp}\sum_{h\in H}x(t,h)\times p(t,h) \quad (5)$$

Subject to:

$$\forall t\in Tp,\ \forall h\in H \quad x(t,h)\in\{0,1\} \quad (6)$$

$$\forall t\in T_p,\quad \sum_{h\in H}x(t,h)\le 1 \quad (7)$$

$$\forall h \in H, \quad \sum_{t \in \mathrm{T}p} r(t) \times x(t,h) + \sum_{t \in \mathrm{T}_h} r(t) \le R(h) \quad (8)$$

$$\forall t \in T_p \wedge h \in H \wedge x(t,h) = 1, e(t,h) \le d(t) \quad (9)$$

This is a multi-dimensional 0-1 knapsack problem, and it is hard to find a solution by polynomial time algorithms.

IV. Task Scheduling Algorithm

To find an sub-optimal solution for the problem defined above, an algorithm is designed in this paper based on ACO which has been adopted to tackle complex combinatorial optimization problems [9][13][18-20].

4.1 Solution representation

Given a specific scheduling timeslot, N tasks are stored in the pending list and wait to be scheduled. Assume there are M available hosts and then, the solution that formulated by the mapping relationship between tasks and hosts in a matrix:

$$X^{M \times N} = \begin{bmatrix} x(1,1) & x(1,2) & \cdots & x(1,N) \\ x(2,1) & x(2,2) & \cdots & x(2,N) \\ \vdots & \vdots & \ddots & \vdots \\ x(M,1) & x(M,2) & \cdots & x(M,N) \end{bmatrix} \quad (10)$$

For each element in this matrix, it represents the link between a task and a host. In order to describe briefly, the task will be considered as the task on the link, and the profit of executing the task on the host will be called the profit of the link. For instance, task t is said to be on link $x(h,t)$ and $p(t,h)$ is the profit of link $x(h,t)$.

4.2 Pheromone value placement and update

In general, existing scheduling problems that considered in [21] and [22] are two instances of the Subset Selection Problem. In other words, the objective is to choose the best subset from the available set with the aim to maximize or minimize the evaluation function. Therefore, in order to improve the total profits, the pheromone is placed on the tasks.

In this paper, the heterogeneity of the hosts are taken into consideration. Considering that the profits obtained from the same tasks with different hosts are varies, the pheromone should be placed on the links between tasks and hosts. Therefore, the target in our scenario is to select the best links set from all the available links.

After the initialization step, the pheromone on each link needs to be updated at the end of each solution searching cycle. Generally speaking, the update process mainly consists of two parts: At first, the pheromone value on each link should be reduced by a certain percentage in corresponding to the real behavior of pheromone evaporation as time goes on. Then, the increment of pheromone value that is laid by the solutions in the last cycle needs to be added on the links. To be specific, the pheromone value on link $l(j, k)$ at time t_1 is assumed to be $\tau_l(t_1)$. Then, at the next update time t_2, the value should be updated to be $\tau_l(t_2)$, which can be calculated by:

$$\tau_l(t_2) = (1-\rho)\tau_l(t_1) + \Delta\tau_l(t_1,t_2) \quad (11)$$

where $0 \le \rho \le 1$ is a coefficient which represents pheromone evaporation, and $\Delta\tau(t_1,t_2)$ represents the increment of pheromone that is laid by the solution derived by the ants in the last cycle:

$$\Delta\tau_l(t_1,t_2) = \sum_{i=1}^{q} \Delta\tau_l^i(t_1,t_2) \quad (12)$$

where q is the number of the ants, $\Delta\tau_l^i(t_1,t_2)$ is the pheromone value laid on link l by ant i's solution at time t_2, which can be calculated by:

$$\Delta\tau_l^i(t_1,t_2) = \begin{cases} G(f(S_i(t_2))) & \text{if ant } i \text{ incorporates link } l \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

where $S_i(t_2)$ is an ant i's solution at time t_2, and $f(S_i(t_2))$ is the value of its evaluation function. In order to maximize the total profits, the evaluation can be calculated by:

$$f(S_i(t_2)) = \sum_{l \in S_i(t_2)} p(j,k) \quad (14)$$

where $p(j,k)$ is the profit that is laid on link l. Thus, $f(S_i(t_2))$ is the total profits which belong to $S_i(t_2)$. The function G in (13) depends on the formulated problem. In our analysis, it can be defined as $G(f(S_i(t_2))) = Q \cdot f(S_i(t_2))$,

where Q is a parameter of the method.

4.3 Local heuristic value

In the process of ACO, local heuristic value is usually defined to accelerate the searching process in corresponding to the positive feedback of ACO. Apart from the profits obtained from the tasks, the resources consumed by the hosts should be taken into consideration while determining the local heuristic scheme. It means that those links which can bring higher profit with less resource consumption will be preferred by the scheduling algorithm.

The resources in host h that are consumed by the partial solution $S_i(t_2)$ of ant i at time t_2 is assumed to be $\mu_h(i,t_2)=\sum_{l\in S_i(t_2)} r_l$, where r_l represents the total resources consumed by task j. The remaining resources of host h can be represented as $\gamma_h(i,t_2)=R(h)-\mu_h(i,t_2)$. Therefore, the tightness of link l on host h can be defined as:

$$\delta_{lh}(i,t_2)=\frac{r_l}{\gamma_h(i,t_2)} \tag{15}$$

i.e., the ratio between r_l, the amount of host h's resource consumed by the task on link l, and $\gamma_h(i,t_2)$. When multi-dimensional resources of hosts should be taken into consideration, the average value of tightness can be calculated and then, it will be assigned to $\delta_{lh}(i,t_2)$.

The average tightness on all providers in case of link l being chosen to be included in $S_i(t_2)$ is:

$$\bar{\delta}_l(i,t_2)=\frac{\sum_{h\in H}\delta_{lh}(i,t_2)}{|\mathrm{H}|} \tag{16}$$

When taking the profits and resources requirement of link l into consideration, the local heuristic value $\eta_l(t_2)$ can be calculated by:

$$\eta_l(t_2)=\frac{p(j,k)}{\bar{\delta}_l(i,t_2)} \tag{17}$$

From (17), we know that a task will be scheduled with a higher probability when it can bring high profit while consuming less resources.

4.4 Link scheduling probability

The probability that link l will be chosen as the next scheduling link of $S_i(t_2)$ is determined by its pheromone value and local heuristic value. The calculation of this probability is:

$$P_l^i(t_2)=\begin{cases}\dfrac{[\tau_l(t_2)]^\alpha[\eta_l(t_2)]^\beta}{\sum_{k\in allowed(t_2)}[\tau_k(t_2)]^\alpha[\eta_k(t_2)]^\beta}, \\ \qquad l\in allowed(t_2) \\ \qquad 0\end{cases} \tag{18}$$

where $allowed(t_2)$ is the set of all the available links. It can be concluded form formula (18) that the bigger the pheromone value and local heuristic value of the link are, the higher the probability it will be chosen.

4.5 Bulletin board and Tabu list

A bulletin board is utilized to remember the best solution, with which the solutions of the ants can compare with it after a new cycle. If one solution is better than the best one in the bulletin board, the optimal solution will be updated according to the new solution. Besides, a tabu list is defined to record the scheduled links in order to avoid to schedule the same link repeatedly.

4.6 Algorithm description

The description of the algorithm is shown in Algorithm 1.

Several parameters including scheduling results (*best_solution*) and its profit (*best_profit*) are initialized in Step 1. The available links that determined by the tasks' resources requirements and available resources on hosts are calculated in Step 4. Step 5 puts the initial pheromone on all the available links. The chosen link based on (18) is decided in Steps 6-9, and the available links are updated in Step 10. When all the available links have been scheduled, the total profits based on the local partial solutions are calculated in Steps 12-16, and If it is larger than the optimal solution in the bulletin, the bulletin will be updated according to the newly optimal solution. After each cycle, the pheromone on each link should be updated based on (13). Besides, a tabu list is employed to help each ant avoid selecting the same links more than once.

For the simplicity of expression, this algo-

rithm will be called DATS-ACO in the simulation section.

The time complexity of the DATS-ACO algorithm is $O(NMCq)$, where C is the number of iterations, and q is the number of ants. Moreover, the space complexity of the algorithm is $O(NM)$.

4.7 Host dynamic handling

An important difference between our cloud-fog environment and traditional cloud-only setting is that the end user IoT devices have relatively high dynamic nature due to their limited energy supply and unreliable wireless links. Note that an IoT device may leave the system without noticing the cloudlet entity due to network disruption or energy supply running out. This type of system churn may hurt the scheduling process and thus the profit of the system.

To tackle this problem, the cloudlet has to judge each IoT device's leaving possibility based on its link quality and remaining energy supply. The cloudlet can put more profitable tasks on those reliable devices with low leaving possibilities. After detecting a leaving IoT device, the cloudlet has to put the unfinished tasks running on the device to the task queue for future scheduling if they have not missed their deadlines.

To be specific, the cloudlet estimates a leaving probability for each IoT device based on their link and energy supply condition. A leaving factor $L(h)$ is defined for each host as:

$$L(h)=(1-q(h))\,(1-e(h)) \tag{19}$$

where $q(h)$ is the link quality of host h and $e(h)$ is the remaining energy supply of host h. They are both normalized to be defined in the interval [0, 1]. High value means good link quality and enough energy supply and low leaving factor. Based on this factor, the leaving probability of a host h is defined as:

$$l(h)=\begin{cases}\varphi_1 L(h), & \text{if } L(h)\le 0.4\\ \varphi_2 L(h), & \text{if } 0.4<L(h)\le 0.7\\ \varphi_3 L(h), & \text{if } L(h)>0.7\end{cases} \tag{20}$$

where φ_1, φ_2, and φ_3 are chosen in the interval [0, 1] to reflect different leaving behavior of the host with different link quality and remaining energy supply.

Algorithm 1. Deadline-Aware Task Scheduling Algorithm for a Tiered IoT Infrastructure based on ACO.

1. *best_solution = []*
2. *best_profit = 0*
3. ***For*** *each time slot*
4. ***For*** *each ant in the ant set*
5. *Find feasible link set based on all tasks' resource requirements and deadline constraints, and hosts' available resources*
6. *Place initial pheromone value on every feasible link*
7. ***While*** *feasible link set is not null*
8. *Calculate local heuristic value for every link*
9. *Select one link for scheduling according to (18)*
10. *Add the selected link to the partial solution*
11. *Adjust the feasible link set based on the resource consumption of the selected link*
12. ***End while***
13. *Calculate the total profit of the partial solution*
14. ***If*** *the profit of current ant's partial solution is larger than best_profit*
15. *Set best_profit to be the profit of the partial solution*
16. *Record the partial solution in the best_solution*
17. ***End if***
18. *Calculate the incremental pheromone on each link according to (13)*
19. *Clear the tabu_list for each ant*
20. ***End for***
21. ***End for***
22. ***Return*** *best_solution and best_profit*

Then, we can incorporate this probability into the scheduling process when deciding each link's scheduling probability in (18). Now, (18) can be rewritten as:

$$P_l^i(t_2)=\begin{cases}\dfrac{[\tau_l(t_2)]^\alpha[\eta_l(t_2)]^\beta l(h(l))}{\sum_{k\in allowed(t_2)}[\tau_k(t_2)]^\alpha[\eta_k(t_2)]^\beta l(h(k))}, & l\in allowed(t_2)\\ 0\end{cases} \tag{21}$$

This means that the lower leaving probability the host on a link, the higher the probability that it will be scheduled.

To realize this refinement in Algorithm 1, we only need to *Select one link for scheduling according to (21)* in step 9.

For the ease of comparison, this refined algorithm will be called Refined DATS-ACO (RDATS-ACO) in the following analysis.

V. Simulation and Results

5.1 Experimental settings

To validate the effectiveness of our proposal, a simulator is built and extensive simulations are conducted based on the parameters listed in Table II. A few typical values are adopted for ACO, i.e. $\alpha=\beta=1$, $\rho=0.3$. There is only one datacenter and one cloudlet in the system, and the initial number of IoT devices is set to be 80. 3 IoT devices join the system in each timeslot while each IoT device has its own energy supply and wireless link quality randomly selected in the interval [0, 1], which is used to derive its leaving probability according to (19). The number of simulation timeslots is 50.

The arrival rate of the tasks obeys Poisson distribution with parameter λ. The minimum resource possession and consumption is assumed to be one unit. To model the heterogeneity of the hosts' resources, the amount of computation resource units at each host, i.e. $R_c(h)$ obeys uniform distribution in the interval between min_c and max_c. Moreover, $R_s(h)$ obeys uniform distribution in the interval between min_s and max_s. Similarly, the computation and storage resource requirement of each task, i.e., $r_c(t)$ *and* $r_s(t)$, obey uniform distribution in $[min_{rc}, max_{rc}]$ and $[min_{rs}, max_{rs}]$, respectively. α_d, the reward factor of each IoT device, obeys uniform distribution in the interval $[min_{rf}, max_{rf}]$. Moreover, the deadline of each task, i.e., $d(t)$, is set to be ϕ times the length of a timeslot and ϕ obeys uniform distribution in $[min_d, max_d]$. φ_1, φ_2 and φ_3 in (20) are set to be 0.16, 0.08, and 0.04 respectively. The default values of some parameters are listed in Table II.

Table II. *Simulation parameters.*

Parameters	Meaning	Value
λ	Arrival rate of tasks at each timeslot	varies
min_c /max_c	Minimum/Maximum amount of computational resource of each host	100/500
min_s /max_s	Minimum/Maximum amount of storage resource of each host	100/500
min_{rc}/max_{rc}	Minimum/Maximum amount of computational resource consumption of each task	50/150
min_{rs}/max_{rs}	Minimum/Maximum amount of storage resource consumption of each task	30/100
min_{rf}/max_{rf}	Minimum/Maximum reward factor of each IoT device	0/0.8
min_d/max_d	Minimum/Maximum deadline of each task	2/10
T_D	The delay on the path between the datacenter and the cloudlet	100*ms*
T_C	The delay on the path between an IoT device and the cloudlet	50*ms*

Comparison Benchmark and Metrics. To show the effectiveness of the proposal, First-Come-First-Served (FCFS) algorithm, Min-min algorithm and algorithm, and Improved Max-min algorithm (IMax-min), are implemented on the simulator as the comparison basis [24]. In FCFS, the tasks are scheduled according to their arrival order and for each task the first host which can meet its resource and deadline requirement would be selected. In Min-min, the execution time between tasks and hosts are calculated in advance and the task with minimum execution time will be scheduled to its corresponding host with higher priority. In IMax-min, the expected execution time for each task and host will be calculated in advance and then, the tasks will be scheduled in a descending way according to the expected execution time.

The profits and the guarantee ratios of the scheduling algorithms are the metrics to compare different algorithms. The profit of a scheduling algorithm is the total profits of all its scheduled tasks. The guarantee ratio equals to the number of scheduled tasks divided by the total number of arrived tasks.

5.2 Experimental results

Profits of the algorithms. The total profits for different algorithms are shown in figure 3. We can see that the total profits of these five algorithms increase as the arrival rate of tasks increases from 30 to 100. This is due to the fact that there are more tasks with high profits available for scheduling when λ increases. When the number of tasks is relative small, the resources of hosts have not been fully

exploited. As the number of tasks increases, the increasing rate of profits decreases. This is due to the relative limited resources of hosts. In addition, the profit of IMax-min is smaller than FCFS and Min-min when the number of tasks exceeds a certain number due to that it always chooses the tasks with larger execution time. Among all the scheduling algorithms, RDATSACO always has the largest profit since it can better find those tasks with higher profits and lower resource requirement and can cope with the dynamic nature of the IoT devices.

Guarantee ratio of the algorithms. The guarantee ratios of the five algorithms are shown in figure 4. From figure 4, we know that all five algorithms' guarantee ratios decrease as the increase of the task arrival rates since there are more tasks that cannot be executed. Furthermore, RDATS-ACO always performs better than the other four algorithms.

5.3 Parameter influence

There are a few parameter settings that can potentially influence the five algorithms' performance. Here, we mainly consider the deadline of each task, the amount of computation resource units at each host, φ_1, φ_2, and φ_3 which decide the leaving probability of the hosts.

The influence of the deadline. Figure 5(a) and figure 5(b) shows the profits and the guarantee ratios of the five algorithms when min_d and max_d are set to be 1 and 8 respectively. From figure 5, we can draw two observations. Firstly, RDATS-ACO always performs better than the other four algorithms during this scenario. Secondly, short deadline can reduce the algorithms' profits and task guarantee ratios since more tasks may miss their deadline when waiting for scheduling in the queue.

The influence of the amount of resources. Figure 6(a) and figure 6(b) show the profits and the guarantee ratios of the five algorithms when min_c / min_s and max_c /max_s are set to be 100 and 300 respectively. In contrast, Figure 7(a) and figure 7(b) show the profits and the guarantee ratios of the five algorithms when min_c / min_s and max_c /max_s are set to be 200 and 600 respectively. From figure 6 and 7, we can draw two observations. Firstly, RDATS-ACO always performs better than the other four algorithms during these two scenarios. Secondly, less and more resources can reduce

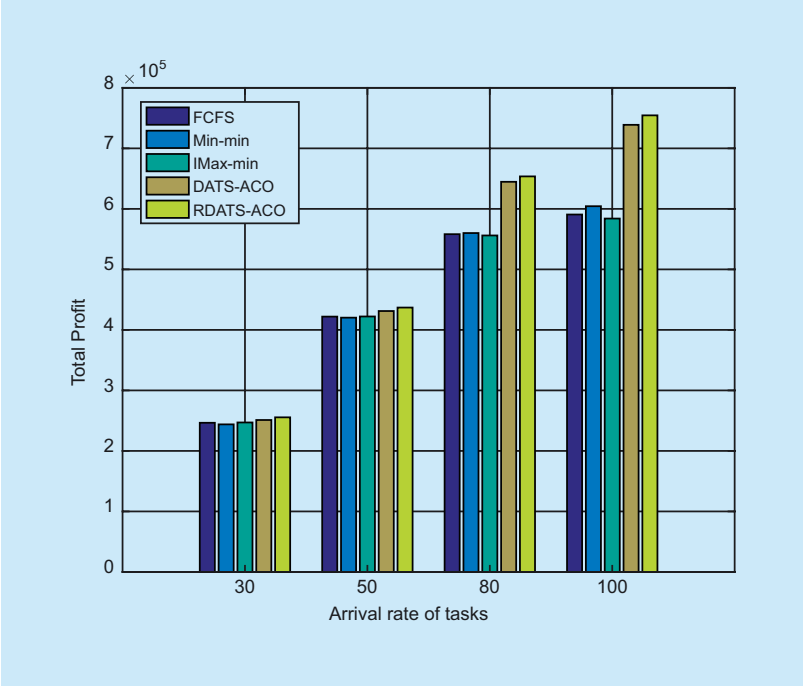


Fig. 3. *The profit of FCFS, Min-min, DATS-ACO, and RDATS-ACO for different arrival rate of tasks.*

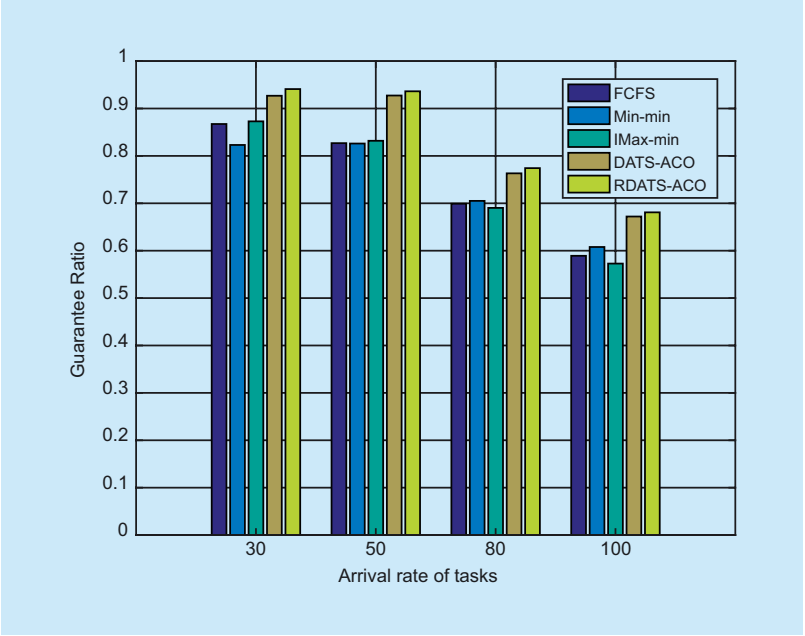
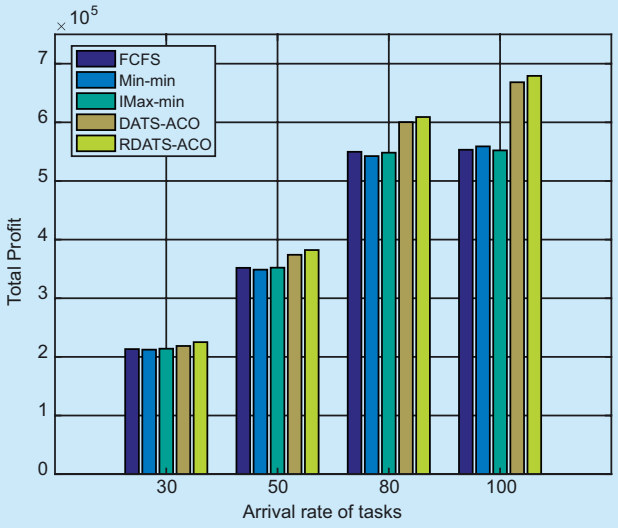


Fig. 4. *The guarantee ratio of FCFS, Min-min, DATS-ACO, and RDATS-ACO for different arrival rate of tasks.*

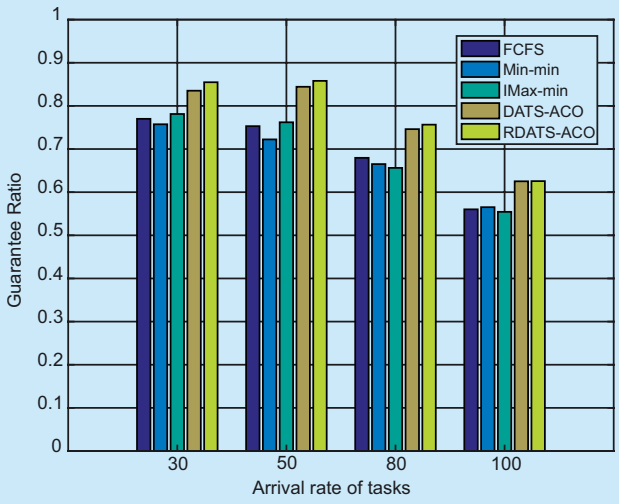
and increase the algorithms' profits and task guarantee ratios respectively.

The influence of the leaving probability. In contrast, Figure 8(a) and figure 8(b) show the profits and the guarantee ratios of the five algorithms when φ_1, φ_2, and φ_3 are all set to be 0.08. Under this setting, hosts with different link and energy supply quality will leave with the same weight. Therefore, the gap between the leaving probabilities of different host groups is narrow and less hosts will leave the system during the process in comparison with the initial setting.

We can see that less leaving probabilities can increase the profit as well as the five algorithms' task guarantee ratios. This is due to

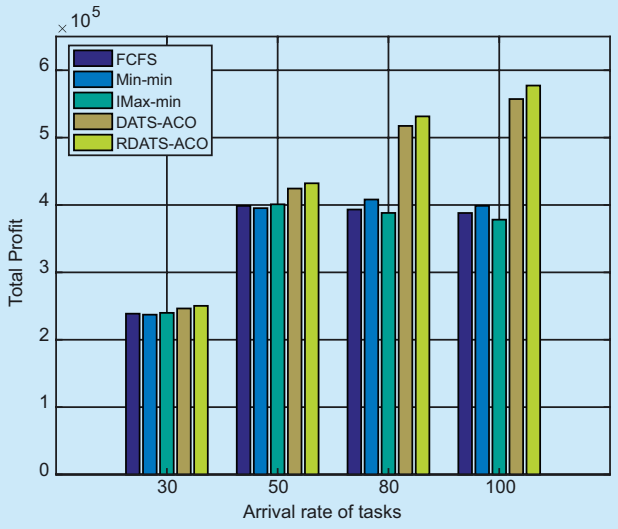


(a) The profit of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

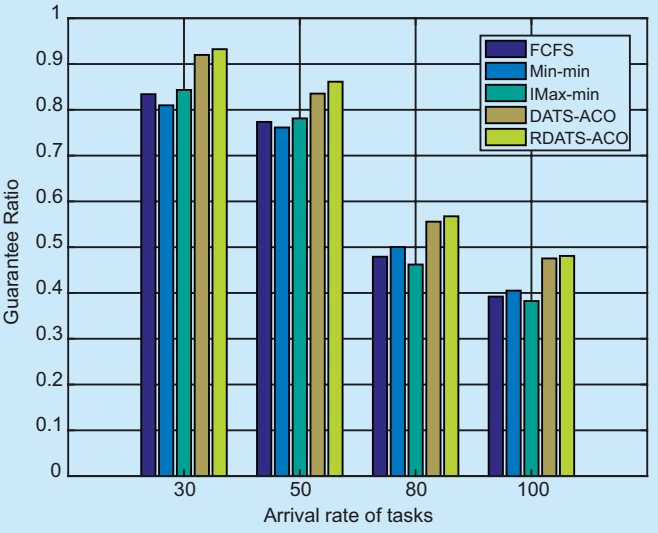


(b) The guarantee ratio of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

Fig. 5. *The profits and the guarantee ratios of the five algorithms when min_d and max_d are set to be 2 and 8 respectively.*



(a) The profit of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.



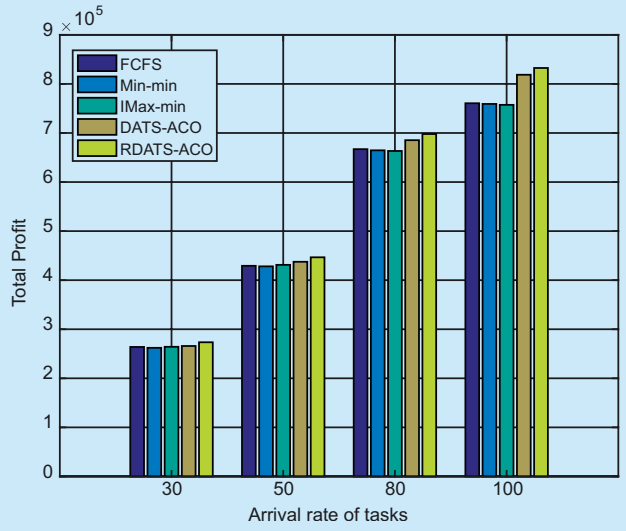
(b) The guarantee ratio of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

Fig. 6. *The profits and the guarantee ratios of the five algorithms when min_c / min_s and max_c /max_s are set to be 100 and 300 respectively.*

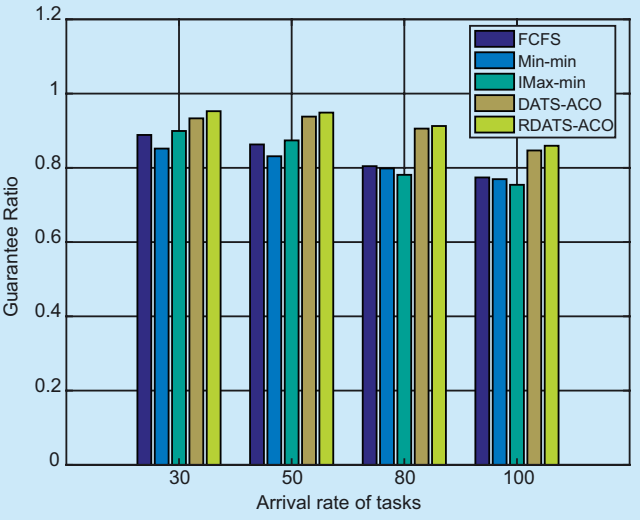
the fact that more resources are available to execute the offloaded tasks when the leaving probability is low.

VI. Conclusion and Future Work

To cope with the challenges brought by tiered Internet of Things (IoT) environments, fog computing is introduced to act as the local computation facility and the relay between IoT devices and the remote data center. To effectively utilize the available resources, this paper formulates the task scheduling problem as a multi-dimensional 0-1 knapsack problem, and put forwards an algorithm based on Ant Colony Optimization (ACO). In our proposal,

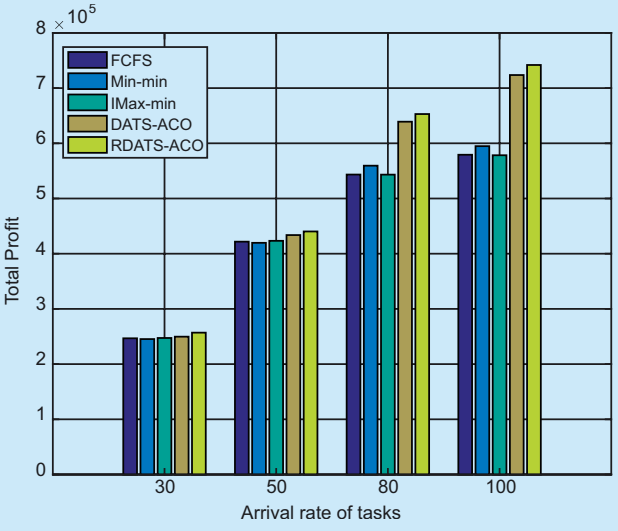


(a) The profit of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

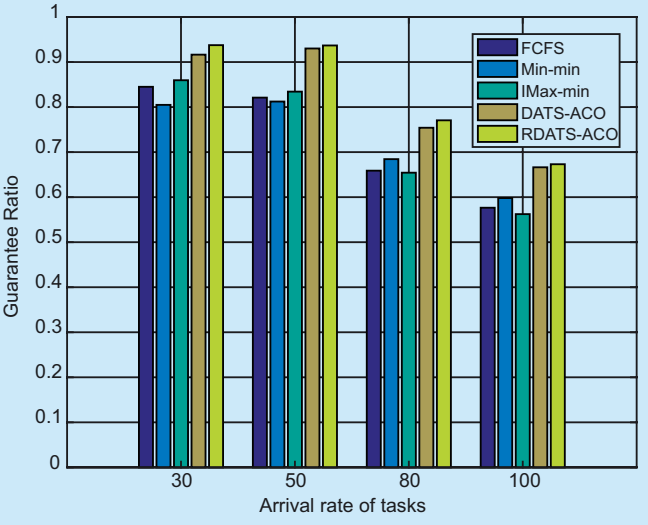


(b) The guarantee ratio of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

Fig. 7. *The profits and the guarantee ratios of the five algorithms when min_c / min_s and max_c /max_s are set to be 200 and 600 respectively.*



(a) The profit of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.



(b) The guarantee ratio of FCFS, Min-min, DATS-ACO, and RDATS-ACO algorithms.

Fig. 8. *The profits and the guarantee ratios of the five algorithms when φ_1, φ_2, and φ_3 are all set to be 0.08.*

the pheromone value is placed on the links between tasks and hosts that execute offloaded tasks, enabling it to maximize the total profits while meeting the tasks' deadlines and resource constraints. To cope with the dynamic nature of the IoT devices, an refined algorithm is presented which takes the hosts' leaving probabilities into consider when scheduling the tasks. Extensive simulations are conducted to evaluate the performance of the proposed algorithms. Numerical results show that our solution outperforms existing three heuristics algorithms. Moreover, the refined algorithm outperforms the original scheduling algorithm since it can better tackle the churn of the system. In the future, we will consider distributed, lightweight algorithms for the joint optimization.

References

[1] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan and G. J. Ren, "Foggy Clouds and Cloudy Fogs: A Real Need for Coordinated Management of Fog-to-cloud Computing Systems," in *IEEE Wireless Communications*, vol. 23, no. 5, 2016, pp. 120-128.

[2] F. Bonomi, R. Milito, J. Zhu, et al., "Fog Computing and Its Role in The Internet of Things", *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, 2012: 13-16.

[3] M. Chiang, T. Zhang, "Fog and IoT: An Overview of Research Opportunities," *in IEEE Internet of Things Journal*, vol. 3, no. 6, 2016, pp. 854-864.

[4] Z. Zhan, X. Liu, Y. Gong, et al., "Cloud Computing Resource Scheduling and A Survey of Its Evolutionary Approaches", *ACM Computing Surveys (CSUR)*, 2015, pp. 63.

[5] Xuan-Qui Pham and Eui-Nam Huh, "Towards Task Scheduling in A Cloud-fog Computing System," *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, Kanazawa, 2016, pp. 1-4.

[6] L. Pu, X. Chen, J. Xu and X. Fu, "D2D Fogging: An Energy-Efficient and Incentive-Aware Task Offloading Framework via Network-assisted D2D Collaboration," in *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, 2016, pp. 3887-3901.

[7] D. Zeng, L. Gu, S. Guo, Z. Cheng and S. Yu, "Joint Optimization of Task Scheduling and Image Placement in Fog Computing Supported Software-Defined Embedded System," in *IEEE Transactions on Computers*, vol. 65, no. 12, 2016, pp. 3702-3712.

[8] Z. Chen, K. Du, Z. Zhan, et al., "Deadline Constrained Cloud Computing Resources Scheduling for Cost Optimization based on Dynamic Objective Genetic Algorithm," *Evolutionary Computation (CEC), 2015 IEEE Congress on*, IEEE, 2015, pp. 708-714.

[9] Z. Chen, Z. Zhan, H. Li, et al., "Deadline Constrained Cloud Computing Resources Scheduling through An Ant Colony System Approach," *Cloud Computing Research and Innovation (ICCCRI)*, 2015 International Conference on. IEEE, 2015, pp. 112-119.

[10] S. Shin, Y. Kim, S. Lee, "Deadline-guaranteed Scheduling Algorithm with Improved Resource Utilization for Cloud Computing," *Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE*, IEEE, 2015, pp. 814-819.

[11] C. Chen, J. Lin, S. Kuo. "MapReduce Scheduling for Deadline-Constrained Jobs in Heterogeneous Cloud Computing Systems," *IEEE Transactions on Cloud Computing*, 2015.

[12] D. Li, C. Chen, J. Guan, et al., "DCloud: Deadline-aware Resource Allocation for Cloud Computing Jobs," *IEEE Transactions on Parallel and Distributed Systems*, 2016, pp. 2248-2260.

[13] D. Komarasamy, V. Muthuswamy, "Adaptive Deadline Based Dependent Job Scheduling algorithm in cloud computing," *Advanced Computing (ICoAC), 2015 Seventh International Conference on. IEEE*, 2015, pp. 1-5.

[14] A. Razaque, N. R.Vennapusa, N. Soni, G. S. Janapati, and K. R. Vangala., "Task Scheduling in Cloud Computing," *in Long Island Systems, Applications and Technology Conference*, 2016, pp.1-5.

[15] X. Yi, F. Liu, Z. Li, et al., "Flexible Instance: Meeting Deadlines of Delay Tolerant Jobs in The Cloud with Dynamic Pricing," *Distributed Computing Systems (ICDCS), 2016 IEEE 36th International Conference on*, IEEE, 2016, pp. 415-424.

[16] L. Zuo, L. Shu, S. Dong, et al., "A Multi-objective Hybrid Cloud Resource Scheduling Method Based on Deadline and Cost Constraints," *IEEE Access*, 2016.

[17] S. Sarkar, S. Chatterjee, S. Misra, "Assessment of The Suitability of Fog Computing in The Context of Internet of Things," *in IEEE Transactions on Cloud Computing*, vol.312, no. 2C3, 2003, pp.266-269.

[18] M. Aazam, E. N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, Gwangiu, 2015, pp. 687-694.

[19] C. Tang, X. Wei, S. Xiao, W. Chen, W. Fang, W. Zhang, and M. Hao. "A Mobile Cloud based Scheduling Strategy for Industrial Internet of Things." *IEEE ACCESS*, vol.6, 2018, pp. 7262-7275.

[20] Y. Yang, K. Wang, G. Zhang, X. Chen, X. Luo, and M. T. Zhou, "Meets: Maximal Energy Efficient Task Scheduling in Homogeneous Fog Networks," *IEEE Internet of Things Journal*, 2018, pp. 1-1.
[21] M. Dorigo, L. Gambardella, "Ant Colony System: A Cooperative Learning Approach to The Traveling Salesman Problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, 1997, pp. 53-66.
[22] X. Wei, J. Fan, T. Wang, et al., "Efficient Application Scheduling in Mobile Cloud Computing based on MAX-MIN Ant System," *Soft Computing*, vol. 20, no. 7, 2016, pp. 2611-2625.
[23] X. Wei, J. Fan, Z. Lu, et al., "Application Scheduling in Mobile Cloud Computing with Load Balancing," *Journal of Applied Mathematics*, 2013, pp. 337-366.
[24] O. Elzeki, M. Reshad, and M. Elsoud, "Improved Max-min Algorithm in Cloud Computing," *International Journal of Computer Applications*, vol. 50, no. 12, 2015, pp. 22-27.
[25] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-aware task scheduling in a tiered IoT infrastructure," *In GLOBECOM 2017-2017 IEEE Global Communications Conference* . IEEE, 2017, pp. 1-7.

Biographies

Jianhua Fan, received the Ph. D. degree from Institute of Communication Engineering, Nanjing, China in 1999. Now he is working as a senior researcher at Nanjing Telecommunication Technology Research Institute, Nanjing, China. His research interests include wireless ad hoc network, cognitive network, and mobile edge computing.

Xianglin Wei, received his bachelor's degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China in 2007, and Ph.D. degree from University of Science and Technology, Nanjing, China in 2012. He is currently a Researcher with the Nanjing Telecommunication Technology Research Institute, Nanjing, China. His research interests include cloud computing, wireless network optimization, and distributed system design and optimization. He has served as an editorial member of many international journals and a TPC member of several international conferences. He has also organized a few special issues for many reputable journals.

Tongxiang Wang, received a Bachelor's degree from the University of Science and Technology, Nanjing, China in 2012. He is now a Ph. D. student in Nanjing Telecommunication Technology Research Institute. His research interests include mobile edge computing and wireless network optimization.

Tian Lan, received the B.A.Sc. degree from the Tsinghua University, China in 2003, the M.A.Sc. degree from the University of Toronto, Canada, in 2005, and the Ph.D. degree from the Princeton University in 2010. Dr. Lan is currently an Associate Professor of Electrical and Computer Engineering at the George Washington University. His research interests include cloud resource optimization, mobile networking, storage systems and cyber security. Dr. Lan received the 2008 IEEE Signal Processing Society Best Paper Award, the 2009 IEEE GLOBECOM Best Paper Award, and the 2012 INFOCOM Best Paper Award.

Suresh Subramaniam, received the Ph.D. degree in electrical engineering from the University of Washington, Seattle, in 1997. He is a Professor in and Chair of the Department of Electrical and Computer Engineering at the George Washington University, Washington DC. He directs the Lab for Advanced Networking and Computing (LINC) at GWU. His research interests are in the architectural, algorithmic, and performance aspects of communication networks, with current emphasis on optical networks, cloud computing, data center networks, and IoT. He has published around 200 peer-reviewed papers in these areas, and has co-edited 3 books on optical networking. He has served as TPC Chair for several conferences including INFOCOM, Globecom, and ICC. He has served on the editorial boards of 7 journals including the IEEE/ACM Transactions on Networking and the IEEE/OSA Journal of Optical Communications and Networking. During 2012 and 2013, he served as the Chair of the IEEE ComSoc Optical Networking Technical Committee, and he is an IEEE ComSoc Distinguished Lecturer for 2018-19. He received the 2017 SEAS Distinguished Researcher Award at GWU. He is a Fellow of the IEEE.