Deadline-Aware Task Scheduling in a Tiered IoT Infrastructure

Jianhua Fan, Xiangli Wei, Tongxiang Wang
Nanjing Telecommunications Technology Institute
Nanjing, China
fjh7659@163.com, wei_xianglin@163.com

Tian Lan, Suresh Subramaniam
George Washington University
Washington DC, US
tlan@email.gwu.edu, suresh@email.gwu.edu

***Abstract*—With the proliferation of the Internet of Things (IoT), the current "*cloud-only*" architectures cannot efficiently handle IoT's data processing and communications needs, while providing satisfactory service latency to support emerging mobile applications on the horizon that require almost real-time responses. fog computing is introduced as a new computing paradigm that distributes computation, communication, control, and storage closer to the end users along the "*cloud-to-things*" continuum. In this paper, we present a deadline-aware task scheduling mechanism for fog computing in a tiered IoT infrastructure, where service providers exploit the collaboration between their own fog nodes and the rented cloud resources to efficiently execute users' offloaded tasks, at large geographical scale. We first formulate the task-scheduling problem in such a cloud-fog environment as a multi-dimensional 0-1 knapsack problem that is NP-hard, and then propose an efficient algorithmic solution based on ant colony optimization heuristic. The main objective is to maximize the profits of fog service provider while meeting the tasks' deadline constraint. Extensive experimental results show that our proposed optimization and solution significantly improves the system performance compared with existing heuristics.**

Keywords—Fog Computing, Task Scheduling, Deadline Constrained, Internet of Things, Ant Colony System

I. INTRODUCTION

As smart cities, smart transportation, and smart homes are transitioning toward the real world, cloud computing, which employs computation resources located far away from the end users and relies on robust network infrastructure, falls short on providing low-latency guarantees to deadline-sensitive applications that are very common in Internet of Things (IoT) scenarios [1].

To address this issue, fog computing (also known as Edge Computing) extends the cloud computing paradigm to the edge of the network, in close proximity to end devices that serve as both data generator and consumer. Many characteristics of fog computing make it the appropriate platform to support critical IoT services and applications. By placing fog resources (i.e., local computing infrastructures) within one-hop of the IoT devices, fog computing can bring a number of key advantages including: a) Low service latency and location awareness; b) Wide-spread geographical distribution and coverage; c) Mobility; d) Scalability with respect to system/network scale, e) Ubiquitous wireless access, f) Strong support for real-time and ultra-low- latency applications, g) Heterogeneity and diversity [2]. A typical architecture of fog computing is shown in Fig. 1. The cloudlet entities act as the local fog computing platforms, connecting to the remote datacenters through the Internet. IoT devices, residing on the network edge, connect to the cloudlet through a wireless network (e.g., cellular or WiFi) and can offload their computation tasks to the cloudlet or the datacenters, if the desired application deadlines can be met.

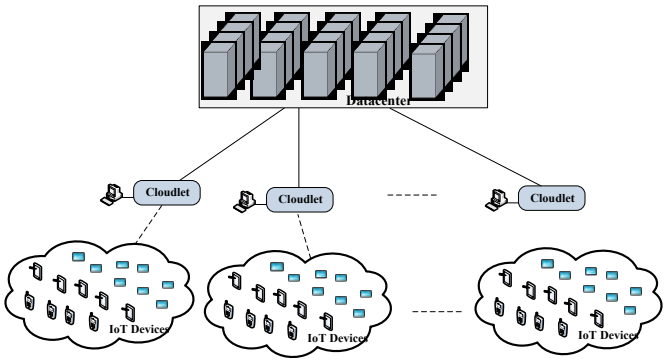

Figure 1. Architecture of a tiered IoT system including cloudlets for edge computing, datacenters for cloud computing, and IoT devices.

Harnessing both fog and cloud computing can enable more agile and context-aware services since the cloudlet entities and the datacenter usually have much more resources than IoT devices and are able to more rapidly execute computation-intensive tasks. Moreover, placing computing resources at the edge of the network allows fog nodes to efficiently process latency-sensitive tasks in a timely manner, while large-scale and latency-tolerant tasks can still be efficiently processed by the cloud that is equipped with more computing power. Many IoT applications such as big data analytics [3] require the interplay and cooperation between the edge (fog) and the core (cloud), calling for a joint optimization of both fog and cloud resources in the network.

In this paper, we present a deadline-constrained task scheduling framework for IoT systems with a joint fog and cloud computing architecture. Our goal is to maximize the total net profit received by service providers through task scheduling and placement, while meeting application deadline

requirements and satisfying resource capacity constraints in both fog and cloud networks. A fog service provider can exploit the collaboration between its own fog nodes (including cloudlet and participating mobile user devices) and the rented cloud resources for efficiently executing tasks offloaded by end users' IoT devices. We show that the proposed task scheduling problem in fog system can be formulated as a multi-dimensional 0-1 knapsack problem, which is known to be NP-hard. Then, a scheduling algorithm based on Ant Colony Optimization (ACO) is presented to maximize the total net profit for the fog service provider, which is defined as total revenue minus execution cost. Finally, we conduct extensive simulations to evaluate the performance of the proposed algorithm. Significant improvement over existing heuristic solutions is demonstrated.

II. Related Work

Fog computing is a newly introduced paradigm, therefore, the number of task scheduling mechanisms specifically aiming at fog computing is quite limited so far [4]. Although there has been a lot of work on task scheduling for cloud computing, we are moving from a centralized cloud model to a distributed heterogeneous fog model, and it will be significantly more complex from the IoT application developer's perspective [3].

In [5], the authors consider task scheduling in a cloud-fog computing system, and propose a heuristic-based algorithm, whose major objective is achieving balance between the makespan and the monetary cost of cloud resources.

In [6], a novel mobile task offloading framework based on network-assisted D2D collaboration is presented, where mobile users can dynamically and beneficially share the computation and communication resources among each other via the control assistance by the network operators. The optimization problem formulationaims to minimize the time-average energy consumption for task executions of all users.

In [7], a fog computing-supported software-defined embedded system is considered. The objective is to balance the workload on a client device and computation servers.

The above papers on task scheduling for fog computing do not consider job deadlines, which are becoming increasingly more important and impact the Quality of Service (QoS).

In [8-16], various deadline-constrained task scheduling mechanisms are studied by some prominent researchers. However, they only focus on cloud computing.

In [17], some comparative analysis and experiments of the fog computing paradigm and the conventional cloud computing paradigm in the context of the tiered IoT system have been conducted. The work shows that in the context of IoT with a large number of latency-sensitive applications, fog computing outperforms cloud computing.

III. Problem Formulation

A. Task Offloading

In the architecture shown in Fig. 1, the IoT devices can offload their tasks to the cloudlet entities or the datacenter. We consider a joint scheduling of all tasks at a given time over five different scheduling choices:

- Execute it locally, i.e. in the cloudlet;
- Transfer and execute it in remote datacenter;
- Execute it in some suitable IoT device;
- Buffer the task in a queue until the next scheduling interval, in which a new optimization is performed;
- Reject the task, so it will no longer be executed by the fog system.

The optimization jointly schedules all arrived and buffered tasks and also determines their placement in the fog and cloud network, i.e., the cloudlet or datacenter executing the task. The scheduling process is illustrated in Fig. 2, where newly arrived tasks submitted by IoT end-users in an optimization interval are jointly optimized together with tasks kept in the buffer. Some of the tasks that are not scheduled for the next interval will be placed in the buffer. The optimization is performed repeatedly to enable online task scheduling and processing.

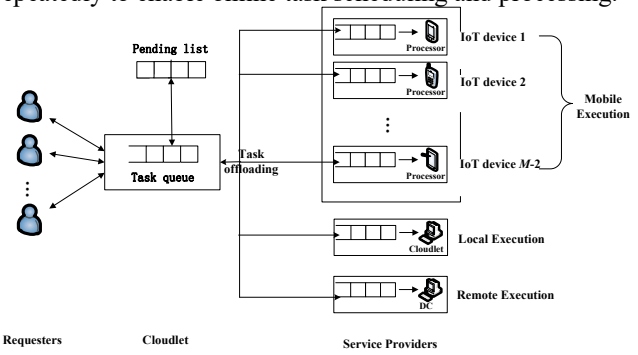

Figure 2. The scheduling process of a tiered IoT system

B. Assumptions and Notations

We make several assumptions in our system model. First, while located far away from IoT end-devices (i.e., incurring high network latency), the datacenter has sufficient resources to process any tasks it is assigned. On the other hand, the cloudlets are only one hop away from the IoT end-devices (i.e., incurring low network latency), but have limited resources for computation. Second, IoT end-devices with underutilized computation, storage, and communication resources can also behave as edge servers to process tasks offloaded by other devices. Each edge or cloud server, i.e., an IoT device (or fog node), the datacenter or the cloudlet, can run multiple tasks concurrently, for example, through resource virtualization. These edge or cloud entities are collectively referred to as hosts in this paper. Next, we assume that each task submitted by IoT end-devices has known resource requirements. In practice, the information can be estimated from previous execution traces or via task profiling. Further, each task has a time-dependent profit function and a hard deadline, meaning that the revenue received by service provider decreases over time and drops down to zero at the deadline. Thus, a task is kept in the buffer until it is rejected by the system after missing the deadline. Finally, for tractability, we consider a time-slotted system

model and focus on scheduling of indivisible tasks. The key notations used in this paper are summarized in Table I.

TABLE I. SUMMARY OF NOTATIONS FOR PROBLEM FORMULATION

Notation		Description
T	T_p	the pending tasks set
	T_h	the current running task set on host h
h	h_{user}	IoT device in the fog system
	$h_{cloudlet}$	cloudlet hosts in the fog system
	h_{cloud}	Hosts in the datacenter
H		Hosts set, $H=\{h_{user}\}\cup\{h_{cloudlet}\}\cup\{h_{cloud}\}$
$R(h)$		The amount of resources of the host h, including computation, communication and storage resources, $R(h)=(R_c(h),R_b(h),R_s(h))$
t		A task
M-2		# of IoT devices in the fog system.The total number of hosts is M, including one cloudlet and one cloud datacenter.
$r(t)$		Resource required by task t, including computation, communication and storage resources, $r(t)=(r_c(t),r_b(t),r_s(t))$
$DS(t)$		Task t's dataset size
T_0		The execution time for processing one unit of data based on 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 on the path between the datacenter and the cloudlet
T_C		The delay on the path between an IoT device and the cloudlet
T'_D		The delay jitter on the path between the datacenter and the cloudlet
T'_C		The delay jitter on the path between an IoT device and the cloudlet
$x(t,h)$		It indicates whether task t is scheduled on host h or not and can be 0 or 1
T_S		The current time of scheduling
$a(t)$		The arrival time of task t
$d(t)$		The deadline of task t
$l(t)$		The estimated execution time of task t
$e(t, h)$		The estimated ending time of task t on host h
$m(t,h)$		The estimated transmission time interval of task t from and to host h
α_d		A reward factor of IoT 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

C. Profits and Execution Time Estimation

Task t's profit running on host h can be calculated 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 $b(h)$ is the basic benefit of running a task on host h, and $c(h)$ the basic cost of running a task on host h, and α_d is a reward factor to stimulate the fog system to finish the task as soon as possible, since the smaller $e(t,h)$ is, the higher $p(t,h)$ will be. $e(t,h)$ is decided by three factors, i.e. task t's start time a_t, task t's execution time $l(t)$ and the transmission time $m(t,h)$, expressed as (2). $l(t)$ and $m(t,h)$ are defined 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 $DS(t)$ indicates t's dataset size which needs to be transmitted during the offloading process.

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

Note that T_D is decided by the network connection from the cloudlet to the datacenter and it is usually much larger than T_C.

D. Problem Formulation

The objective of task scheduling is to maximize the profits as well as guarantee the deadline constraint, and the problem can be formulated as (5)-(9).

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

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

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

$$\forall h\in H,\quad \sum_{t\in Tp}r(t)\times x(t,h)+\sum_{t\in T_h}r(t)\le R(h) \quad (8)$$

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

We can see that this is a multi-dimensional 0-1 knapsack problem, and is NP-hard.

IV. TASK SCHEDULING ALGORITHM

In order to solve this problem, this paper proposes a scheduling algorithm based on ACO which has been widely used to solve complex combinatorial optimization problems[9][13][18][19][20].

A. Solution representation

At some particular timeslot, assume that there are in total N tasks for scheduling. Then, each solution obtained by the scheduling algorithm is a matrix:

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

Each element in $X^{M\times N}$ can be seen as a link between a task and a host. For ease of description, the task will be called the task on the link and the profit obtained by executing the task on the host will be called the profit of the link.

B. Pheromone value placement and update

The scheduling problem here is different from the problems considered in [19] and [20], which belong to the Subset Problem, i.e. given a set S of n tasks for scheduling and an evaluation function $f()$, the target is to select the best subset of S to maximize or minimize $f()$. In [19] and [20], pheromone value is placed on the tasks since it makes no difference to execute the same task on different hosts.

However, the heterogeneity of the hosts in our scenario makes this assumption invalid here. Therefore, the pheromone value should be put on the links between the tasks and the hosts. This means that a link with a higher pheromone value can better satisfy the requirements of the evaluation function.

After obtaining solutions, the pheromone on the links will be updated. The update process includes two parts: Firstly, the pheromone value on a link is reduced by a certain percentage to emulate the real-life behavior of evaporation of pheromone count over time. Secondly, the pheromone value increment laid by the new solutions of the ants will be added. Assume the pheromone value on link $l(j, k)$ at time t_1 is $\tau_l(t_1)$, then at the next update time t_2, the value is updated to $\tau_l(t_2)$:

$$\tau_l(t_2)=(1-\rho)\tau_l(t_1)+\Delta\tau_l(t_1,t_2) \tag{11}$$

where $0\le\rho\le 1$ is a coefficient which represents pheromone evaporation, $\Delta\tau(t_1,t_2)$ is the pheromone value increment obtained from all the ants' partial solutions:

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

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, and is defined as:

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

where $S_i(t_2)$ is the solution of ant i at time t_2, and $f(S_i(t_2))$ is the value of the evaluation function of this solution. To maximize the profit, the evaluation is defined as:

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

where $p(j,k)$ is the profit brought by link l. Therefore, $f(S_i(t_2))$ is the total profit of the links belonging to $S_i(t_2)$. The function G in (13) depends on the problem, in this paper, it is defined as $G(f(S_i(t_2)))=Q\cdot f(S_i(t_2))$, where Q is a parameter of the method.

C. Local heuristic value

The positive feedback of the ant colony algorithm is usually combined with some local heuristic scheme to accelerate the search process. Here, the local heuristic scheme needs to consider the profits of the links as well as the resources they consumed.

Let $\mu_h(i,t_2)=\sum_{l\in S_i(t_2)}r_l$ be host h's resource consumed by the partial solution $S_i(t_2)$ of ant i at time t_2, where r_l indicates the amount of resources consumed by task j on link l. Then, $\gamma_h(i,t_2)=R(h)-\mu_h(i,t_2)$ is the remaining amount of resources on host h. The tightness of link l on host h is 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 $\gamma_h(i,t_2)$. If there are multiple types of resources for consideration, we will calculate their average value and assign it 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:

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

In order to consider link l's profit as well as its resource requirement, the local heuristic value $\eta_l(t_2)$ is defined as:

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

This means that those tasks that have higher profits and consume less resources are more likely to be scheduled by the cloudlet.

D. Link scheduling probability

After obtaining the pheromone value and local heuristic value on each link, the probability that l will be selected as the next scheduling link of $S_i(t_2)$ is:

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

where $allowed(t_2)$ is the set of all the schedulable links. From (18) we can see that the more pheromone value and local heuristic value a link has, the higher the probability that it will be scheduled.

E. Bulletin board and Tabu list

In addition, we set a bulletin board to record the best solution up to time t_2, with which each ant can compare its own solution. If its solution is better than the best one, it will update the best one with its solution. Besides, during the searching process, a tabu list is set for each ant to help it avoid scheduling the same link more than one time.

F. Algorithm Description

The scheduling algorithm can be illustrated as following.

Algorithm 1: *Deadline-Aware Task Scheduling Algorithm for a Tiered IoT Infrastructure based on ACO*
1. best_solution = [], best_profit = 0
2. For each scheduling cycle
3. For each ant in the ant set
4. Determine feasible link set based on tasks' resources requirement and hosts' available resources and the deadline constraint of the tasks
5. Place initial pheromone trail value on each feasible link
6. While feasible link set is not null
7. Calculate local heuristic value for each link
8. Select one link for scheduling according to (18)
9. Add the scheduled link to the partial solution
10. Adjust the feasible link set based on the resource consumption of the scheduled link
11. End while
12. Calculate the total profit of the partial solution
13. If the profit of current ant's partial solution is larger than best_profit
14. Set best_profit to be the profit of the partial solution
15. Record the partial solution in the best_solution
16. End if
17. Calculate the incremental pheromone on each link according to (11)
18. Clear the tabu_list for each ant
19. End for
20. End for
21. Return best_solution and best_profit

Step 1 initiates the parameters for scheduling results (*best_solution*) and its profit (*best_profit*). For each ant in each scheduling cycle, Step 4 decides the feasible links based on all the tasks' resource requirements and available resources on each host, and the deadline constraint of the tasks. Step 5 places initial pheromone trail value on each feasible link. Step 6-9 select a link for scheduling according to (18). Based on the selected link, Step 10 adjusts the feasible link set. When no more link is available for scheduling, Step 12-16 calculates the total profit of the current partial solution and if it is larger than the current optimal solution, it will be assigned as the new solution. After this, the pheromone on each link will be updated according to (11). During this process, a tabu list is adopted to help each ant avoid selecting the same links for scheduling.

For ease of description, this algorithm will be referred to as DATS-ACO in the following analysis.

G. Complexity analysis

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 $O(NM)$.

V. Simulation and Results

A. Experimental Settings

In order to evaluate the performance of the proposed algorithm, a series of experiments is carried out and the main parameters are listed in Table II. For the parameters used in ACO, we adopt a few typical values, i.e. $\alpha=\beta=1$, $\rho=0.3$. There is only one datacenter and one cloudlet in the system, and the number of IoT devices is set to be 200, which remains stable during the simulation. Besides, the number of simulation timeslots is set to be 100.

The arrival rate of the tasks obeys Poisson distribution with average value 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, $R_s(h)$ also 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* $r_s(t)$, also 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, where φ obeys uniform distribution in $[min_d, max_d]$. The default values of these 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 computation resource of each host	80/200
min_s /max_s	Minimum/Maximum amount of storage resource of each host	60/200
min_{rc}/max_{rc}	Minimum/Maximum amount of computation 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	1/5
T_D	The delay on the path between the datacenter and the cloudlet	100ms
T_C	The delay on the path between an IoT device and the cloudlet	50ms

Comparison Benchmark and Metrics. The benchmark algorithms adopted are First-Come-First-Served (FCFS) algorithm and Min-min algorithm. 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.

The profit and the guarantee ratio of the scheduling algorithms are chosen as the metrics to compare the algorithms. The profit of a scheduling algorithm is defined as the total profit of all the scheduled tasks by the algorithm. The guarantee ratio equals the number of scheduled tasks divided by the number of arrived tasks.

B. *Experimental Results*

Profits of the algorithms. The total profits for different algorithms are shown in Fig. 3. We can see that the total profits of thes three algorithms increase as the arrival rate of tasks increases. This is due to the fact that there are more tasks available for scheduling when λ increases. Furthermore, DATS-ACO always has the largest profit of the three considered algorithms since it can better find those tasks with higher profit and lower resource requirement.

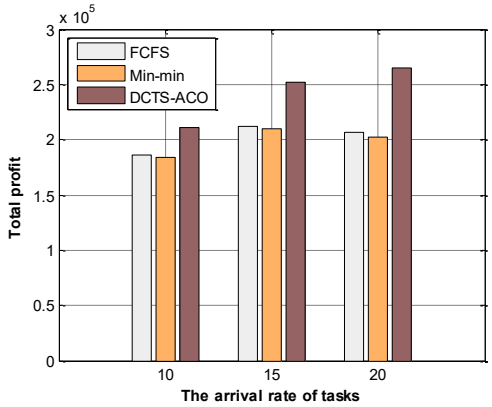

Figure 3. The profit of FCFS, Min-min and DATS-ACO for different arrival rate of tasks.

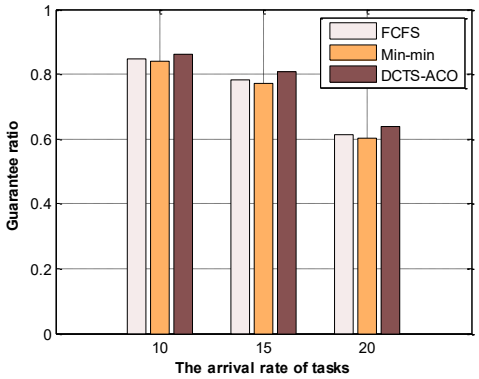

Figure 4. The guarantee ratio of FCFS, Min-min and DATS-ACO for different arrival rate of tasks.

Guarantee ratio of the algorithms. The guarantee ratios for the three algorithms are shown in Fig. 4. We can see that, generally speaking, the guarantee ratios decrease as the arrival rate of tasks increases since there are more tasks that cannot be scheduled. Furthermore, DATS-ACO always performs better than the other two algorithms.

C. *Parameter influence*

The influence of β. As mentioned above, α and β measure the relative importance of the pheromone and local heuristic values. However, local heuristic value is usually much less than the pheromone value. Therefore, increasing β can increase the influence of the local heuristic value on a link's selection probability. Fig. 5 and Fig. 6 show the profit and guarantee ratio, respectively, of the fog system when α=1 and β varies from 1 to 8. Fig. 5 tells us that the total profit first increases with β and then becomes somewhat stable, achieving its peak when β=4, after which it slightly decreases. We can make a similar observation on the curve for the guarantee ratio. Therefore, under our simulation scenario, α=1 and β=4 is a suitable parameter pair.

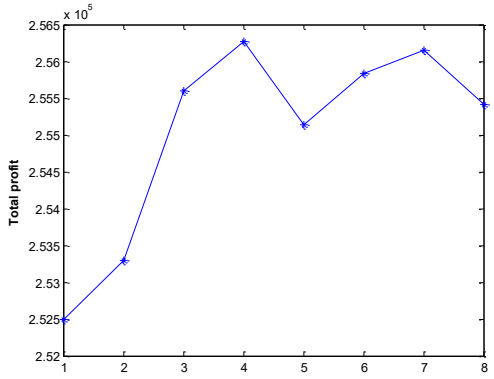

Figure 5. The profit of DATS-ACO when α=1 and β varies from 1 to 8.

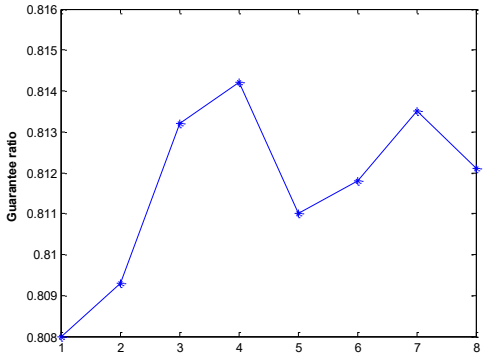

Figure 6. The guarantee ratio of DATS-ACO when α=1 and β varies from 1 to 8.

The influence of T_D. According to (1) and (4), we know that T_D may greatly influence the profit of executing a task on the datacenter and thus the total number of tasks assigned to the datacenter. Here, we evaluate the influence of T_D through setting it to be 1, 2 and 3 times as much as T_C. Generally speaking, the number of tasks offloaded to the datacenter decreases as T_D increases since executing on the datacenter would incur larger latency and thus lower profit. Accordingly, the number of tasks scheduled to be run on the cloudlet and IoT devices increase. Table III shows the profits and guarantee ratios with different settings of T_D.

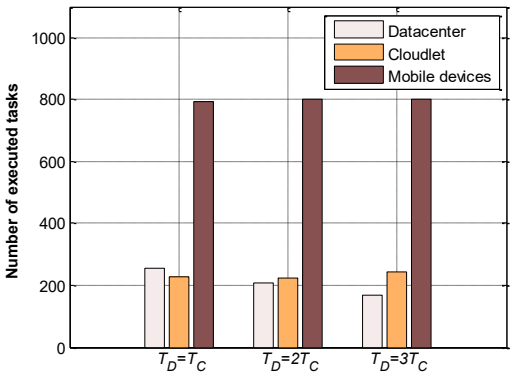

Figure 7. The number of tasks executed on different hosts when T_D is set to be 1, 2, and 3 times T_C.

TABLE III. THE PROFIT OF DATS-ACO WHEN T_D IS SET TO BE 1, 2 AND 3 TIMES AS MUCH AS T_C.

	T_D= T_C	T_D= 2T_C	T_D=3T_C
Total Profit	259850	252490	249250
Guarantee Ratio	0.8355	0.8080	0.7949

VI. CONCLUSION AND FUTURE WORK

This paper formulates the task scheduling problem in a tiered IoT environment as a multi-dimensional 0-1 knapsack problem which is NP hard. An efficient algorithm is proposed based on Ant Colony Optimization (ACO). In our algorithm, the pheromone value is placed on the links between tasks and hosts that execute offloaded tasks, enabling our algorithm to maximize the total net profits while meeting the tasks' deadlines and resource constraints. Extensive simulations are conducted to evaluate the performance of the proposed algorithm. Numerical results show that our solution outperforms existing heuristics including FCFS and Min-min algorithms. In the future, we will consider distributed, lightweight algorithms for the joint optimization.

ACKNOWLEDGMENT

This research was supported in part by NSF grant 1320226, the National Natural Science Foundation of China Grant No. 61402521, and Jiangsu Province Natural Science Foundation of China Grant No. BK20140068 and No. BK20150201.

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, pp. 120-128, October 2016.

[2] Bonomi F, Milito R, Zhu J, et al. Fog computing and its role in the internet of things[C]//Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012: 13-16.

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

[4] Zhan Z H, Liu X F, Gong Y J, et al. Cloud computing resource scheduling and a survey of its evolutionary approaches[J]. ACM Computing Surveys (CSUR), 2015, 47(4): 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, pp. 3887-3901, Dec. 2016.

[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, pp. 3702-3712, Dec. 1 2016.

[8] Chen Z G, Du K J, Zhan Z H, et al. Deadline constrained cloud computing resources scheduling for cost optimization based on dynamic objective genetic algorithm[C]//Evolutionary Computation (CEC), 2015 IEEE Congress on. IEEE, 2015: 708-714.

[9] Chen Z G, Zhan Z H, Li H H, et al. Deadline constrained cloud computing resources scheduling through an ant colony system approach[C]//Cloud Computing Research and Innovation (ICCCRI), 2015 International Conference on. IEEE, 2015: 112-119.

[10] Shin S M, Kim Y, Lee S K. Deadline-guaranteed scheduling algorithm with improved resource utilization for cloud computing[C]//Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE. IEEE, 2015: 814-819.

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

[12] Li D, Chen C, Guan J, et al. DCloud: deadline-aware resource allocation for cloud computing jobs[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(8): 2248-2260.

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

[14] Sidhu H S. Cost-Deadline Based Task Scheduling in Cloud Computing[C]//Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on. IEEE, 2015: 273-279.

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

[16] Zuo L, Shu L, Dong S, et al. A Multi-objective Hybrid Cloud Resource Scheduling Method Based on Deadline and Cost Constraints[J]. 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.PP, no.99, pp.1-1.

[18] M. Dorigo and L. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 53-66, 1997.

[19] Wei X, Fan J, Wang T, et al. Efficient application scheduling in mobile cloud computing based on MAX---MIN ant system[J]. Soft Computing, 2016, 20(7):2611-2625.

[20] Xianglin Wei, Jianhua Fan, Ziyi Lu, and Ke Ding. Application Scheduling in Mobile Cloud Computing with Load Balancing[J]. Journal of Applied Mathematics, 2013, 2013(3):337-366.