Peer-to-Peer Networking and Applications (2020) 13:548–563
https://doi.org/10.1007/s12083-019-00783-7

Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture

Huaiying Sun[1,2] · Huiqun Yu[1,3] · Guisheng Fan[1] · Liqiong Chen[4]

Received: 12 July 2018 / Accepted: 21 June 2019 / Published online: 19 July 2019

Abstract
With the exponential increase in the number of IoT devices and the amount of emitted data from these devices, it is expensive and inefficient to offload all tasks to the remote data center. How to optimize the energy consumption of application requests from IoT devices meeting the deadline constraint is also a challenge. Fog computing adjacent to users has the feature of lower service delay but less resource than the remote cloud. Fog does not appear to replace cloud, they are complementary to each other, cooperation between them is worth studying. This paper proposes a general IoT-fog-cloud architecture that fully exploits the advantages of fog and cloud. Then, the energy and time efficient computation offloading and resource allocation is formulated into the energy and time cost minimization problem. We then propose an ETCORA algorithm to solve the problem, improving the energy consumption and completion time of application requests. Finally, extensive simulations are carried out to verify that the proposed method indeed outperforms the other two methods in reducing energy consumption and completion time of requests.

Keywords IoT · Cloud · Fog · Resource allocation · Completion time · Energy consumption

1 Introduction

Nowadays, the Internet of things (IoT) has become an indispensable part of people's life. It has penetrated into a lot of aspects of our daily life, such as transportation, medical, industrial automation, smart home and emergency response, etc [1]. IoT enables things to see and perceive their environments, make collaborative decisions, and perform corresponding tasks based on the observed data [2]. Currently, a huge amount of data emitted by distributed IoT devices are transferred to the centralized cloud for processing before the results are returned back from the cloud to data consumers, which are always located adjacent to the original data sources [3]. This always incurs high delays and impressive costs for using the cloud-based computational resources [4]. In order to fully realize the role of IoT, there is a need to provide a sufficient network and computing infrastructure to support the requirements of IoT applications for lower delay and faster response [5]. Because of the limited resource on IoT devices, as the number of IoT devices and the observed data grow exponentially, the existing approaches to improve the computing power of IoT devices are to offload the tasks of various application requests (requiring continuous computing) to a computing system with sufficient resource, such as the remote cloud data center (like EC2), which are expensive, yet inefficient [6, 7].

It is not efficient for some applications with rigid service delivery deadlines such as virtual reality, augmented reality and so on to be executed on remote cloud only. A promising

This article is part of the Topical Collection: *Special Issue on IoT System Technologies based on Quality of Experience*
Guest Editors: Cho Jaeik, Naveen Chilamkurti, and SJ Wang

✉ Huaiying Sun
ecustshy@foxmail.com

✉ Huiqun Yu
yhq@ecust.edu.cn

✉ Guisheng Fan
gsfan@ecust.edu.cn

Liqiong Chen
lqchen@sit.edu.cn

1 Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China

2 Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai, China

3 Shanghai Engineering Research Center of Smart Energy, Shanghai, China

4 Department of Computer Science and Engineering, Shanghai Institute of Technology, Shanghai, China

approach is to adopt the decentralized processing of IoT data [2]. In fact, IoT devices (e.g., gateways, sensors, or embedded systems) provide computational, storage, and networking resources, which allow to transfer the execution of IoT applications to the edge of the network [7]. Thus, the approach is described as fog computing (or edge computing) [4]. Fog computing as an extension of the cloud based infrastructure, provides computing, storage, and network resource similar to cloud and promoting the implementation of the IoT application near the data source [8]. Compared with the remote data center, it has the advantage of shortening application response time. Fog nodes are dispersed in geographic location and resource on them are limited compared with cloud. Between fog nodes, network round-trip time, data processing speed and resource availability are still very important [9]. Thus, in a fog computing environment, it is still a challenge to optimize the completion time and energy consumption of a request at the same time.

Therefore, this paper mainly deals with the energy and time efficient computation offloading and resource allocation for IoT applications, which aims at reducing the energy consumption and completion time of IoT application request, improving the computing power of IoT devices. In fog, network entities such as gateway servers, routers, switches, etc. can be regarded as fog nodes for computing purpose [5, 8]. Fog is not to replace the cloud, they are complementary to each other [4]. So the cooperation between fog and cloud is worth studying. We first propose a generic IoT-fog-cloud architecture that makes full use of the advantages of fog (near the user, lower delay) and cloud (far from the user but with sufficient resource). Then, the energy and time efficient computation offloading and resource allocation problem is formulated into the energy and time cost (ETC) minimization problem. To solve this problem, we propose an energy and time efficient computation offloading and resource allocation (ETCORA) algorithm consisting of 2 parts: computation offloading selection and transmission power allocation. Finally, extensive simulation experiments are carried out to verify the effectiveness of the proposed method.

The organization of this paper is as follows: Section 2 introduces the related work. Section 3 introduces the system architecture and computation models. The following is the problem formulation. Section 5 describes the ETCORA algorithm, and the simulation evaluation is shown in Section 6. The last one is the conclusion.

2 Related work

This section describes some works related to this paper [10], and further illustrates the similarities and differences between existing research and our work.

So far, cloud computing has been widely studied based on the elastic and flexible cloud infrastructure, such as the resource/traffic optimization of backhaul networks [11], pricing strategies of using commerce cloud services [12], services admission control [13] and scheduling [14], etc. The resource scheduling schemes can be divided into six hybrid categories containing cost aware resource scheduling, efficiency aware resource scheduling, energy aware resource scheduling, load balancing aware resource scheduling, QoS aware resource scheduling and utilization aware resource scheduling [15]. However, lots of the existing cloud computing models are designed for traditional web applications. In addition, Cloud of Things (CoT) [16] is a very promising computing mode, in which IoT capabilities are retained on demand based services. IoT can overcome its own resource constraint by using virtual unlimited resource in the cloud, cloud can enhance its services by interacting with things in the physical world (such as expanding the scope of implementation) [17]. And IoT analytics also have been largely studied [18–20]. For example, Patel et al. [18] presented a flexible architecture for IoT data analytics using the concept of edge computing that facilitates the automated transitions between edge and cloud depending on the dynamic conditions of the IoT infrastructure and application requirements. Similarly, Daneels et al. [19] depicted an IoT platform that supports reliable and guaranteed data dissemination and analysis (both at the edge and cloud) for rural or remote areas where the wireless infrastructure is sparse, requiring long-range multi-hop connectivity to remote sensors and actuators. Raafat et al. [20] proposed a fog level IoT analysis system that uses an effective knowledge extraction technology to reduce the amount of data that would be transmitted to the cloud and helps to simplify the entire transmission process.

On the other hand, fog computing, characterized by extending cloud computing to the network edge, has become a buzzword today [21, 22]. Jalali et al. [22] proposed a generic fog data flow processing and analysis model and architecture, which is very common in the cloud but has not been fully studied in the fog structure by analyzing the common attributes of various typical applications like the video mining and event monitoring etc. Verma et al. [23] proposed a method for fog-cloud environment, which combines data backup and load balancing based on the availability of data to assign user requests to fog or cloud. This method is more effective than the other cloud technologies such as rotation method, but it is more expensive. Xiao et al. [24] proposed an oFfline Task Assignment (FTA) algorithm and an oNline Task Assignment (NTA) algorithm for mobile crowdsensing. Wen et al. [25] proposed a fog business framework based on parallel genetic algorithm. The framework focuses on the optimization of the selection and placement of fog resources and IoT devices. Yousefpour et al. [5] proposed a method (we call it DMP in this paper) which firstly judges whether the estimated waiting time of a task on the local IoT device will exceed the preset threshold α, if the threshold is exceeded, then a fog node is selected randomly. Similarly, if the waiting time of the task on the fog node exceeds the threshold α, the task will be considered to be forwarded to an adjacent fog node before reaching the largest forwarding number N_{fwd}. If N_{fwd} is exceeded, meaning no fog node can execute the current task, then it will be forwarded to the cloud. Pham et al. [26] proposed a task scheduling algorithm in fog-cloud environment. Experimental results show that the algorithm is suitable for balancing performance and overhead. However, it does not consider the problem of energy consumption.

There are also some studies that are related to the task energy consumption. Chen et al. [27] generally optimize the offloading decisions for all users' tasks and the allocation of computation and communication resources, minimizing the overall energy, computation, and delay cost for all users. Despite the great improvement, it still has to encounter the bottleneck of the limit resources consisting of the local devices. Wang et al. [28] proposed a CachinMobile method to continuously improve the energy efficiency of fog nodes by caching the request data on the terminal user's mobile phone to the fog node. Ni et al. [29] proposed an efficient and secure service-oriented authentication framework supporting network slicing and fog computing for 5G-enabled IoT services. Zhang et al. [30] attempted to develop efficient optimization strategies to save the energy cost of mobile devices for online video streaming over 4G LTE networks. Chang et, al. [8] proposed an approach (we call it ADMMD in this paper) which is to solve the energy consumption optimization problem that meets the delay requirement. Unlike this paper, ADMMD does not consider the dependencies between tasks in the application request and the cooperation between fog and cloud to reduce the energy consumption and completion time of requests. Fog is not to replace the cloud, but to make up for the cloud, the two are complementary to each other. So this paper is concerned with the cooperation between fog and cloud to minimize the completion time and energy consumption of application request, improving the computing power of IoT device.

Table 1 Notations

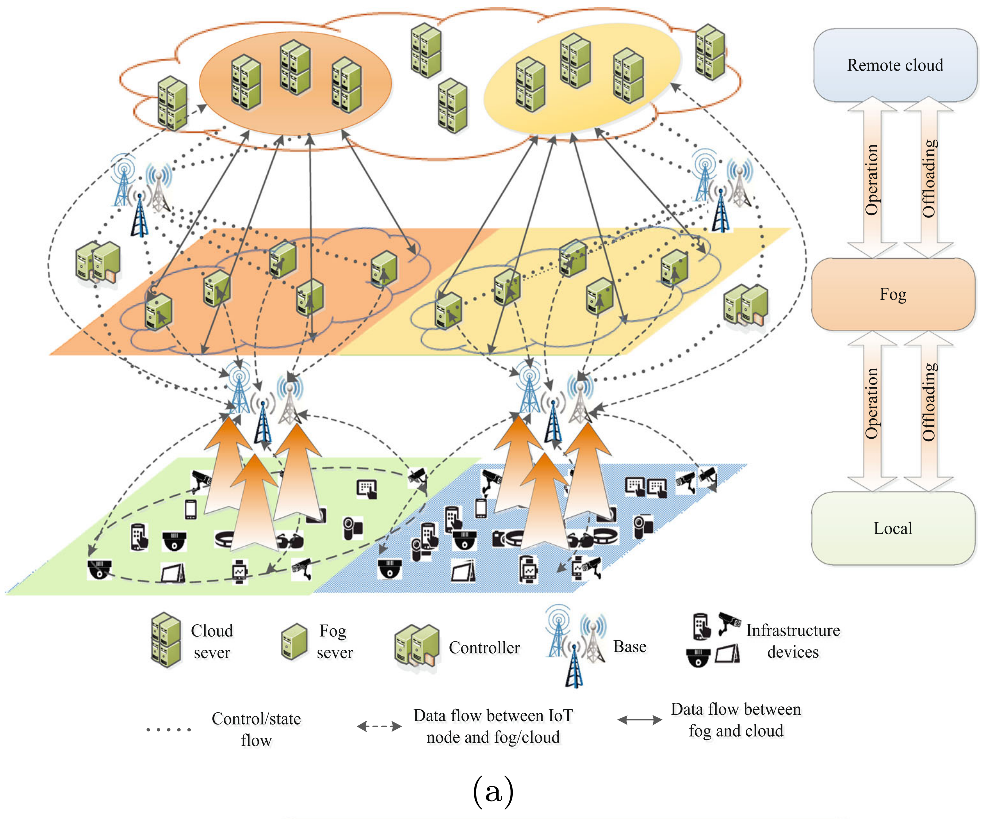
Notations	Definition
C	The number of wireless channels
$TP_{n,m}$	Transmission power for the task m of request n
$CN_{n,m}$	The channel gain between the device n and base station s for task m
$b_{n,m}$	The input data size of task m of request n
$d_{n,m}$	The total number of CPU cycles required to
$r_{n,m}$	The data transmission rate for the task m of request n
$f^l_{n,m}$	The computing capability of the local device for the task m of request n
$f^f_{n,m}$	The computing capability of the fog node for the task m of request n
$f^c_{n,m}$	The computing capability of the cloud node for the task m of request n
$t^l_{n,m}$	The execution time for task m of request n on the local device
$E^l_{n,m}$	The energy consumption for task m of request n on the local device
$E^f_{n,m}$	The energy consumption for task m of request n on the fog
$E^c_{n,m}$	The energy consumption for task m of request n on the cloud
$T^{f,trs}_{n,m}(S)$	The requiring transmission for task m of request n to be offloaded to the fog
$T^{c,trs}_{n,m}(S)$	The requiring transmission for task m of request n to be offloaded to the cloud
$E^{f,trs}_{n,m}(S)$	The requiring energy consumption for task m of request n to be offloaded to the fog
$E^{c,trs}_{n,m}(S)$	The requiring energy consumption for task m of request n to be offloaded to the cloud
$t^{f,exe}_{n,m}$	The execution time of task m of request n on the fog
$t^{c,exe}_{n,m}$	The execution time of task m of request n on the cloud
$CT^l_{n,m}$	The completion time of task m on the local device
$CT^f_{n,m}$	The completion time of task m on the fog
$CT^c_{n,m}$	The completion time of task m on the cloud
$\lambda^t_{n,m}, \lambda^e_{n,m}$	The weights for the computation time and the energy consumption

3 System architecture and computing models

This section introduces the IoT-fog-cloud architecture, communication model and the computation models on the three layers. The commonly used notations in this paper are shown in Table 1.

3.1 System architecture

Figure 1 shows a general architecture of IoT-fog-cloud. The architecture has three layers. The first layer is the infrastructure layer, which contains some IoT devices. The second layer is the fog layer, including some fog servers and controllers, which can be located at different geographic



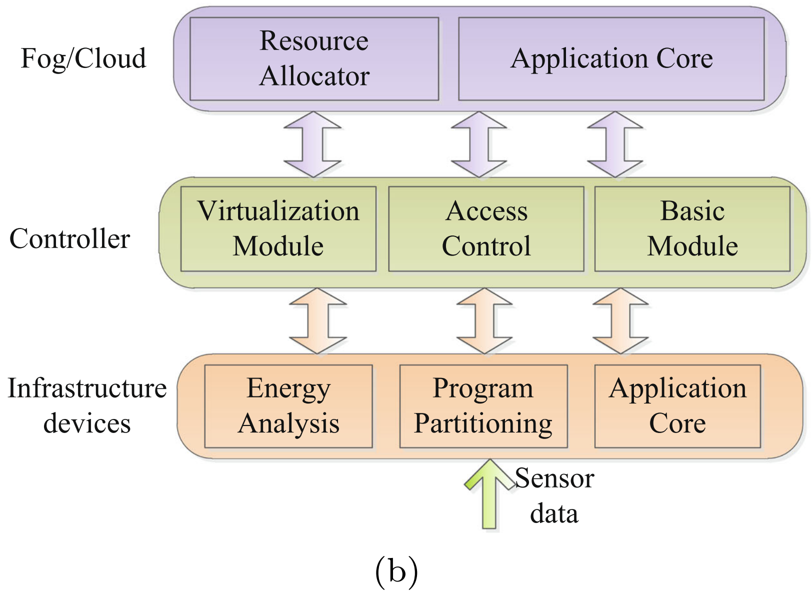


Fig. 1 The system architecture

locations (such as the nearby university or building) [5]. They could be some not fully utilized servers or some gateway servers, routers and switches with computing capability, etc [4]. The third layer is the cloud layer comprising cloud servers. The nodes in the each layer are partitioned into multiple domains, and each domain only realizes a certain IoT application. The nodes in a domain of the first layer (such as the green area of Fig. 1a) can communicate with the fog and cloud nodes (in the orange areas) which implement the same IoT application, for example, offloading one task or some tasks of the corresponding application request to the fog or the cloud belonging to the same domain.

Cloud and fog nodes can be connected to base stations through LTE link or wired backhaul [9]. IoT nodes can send QoS requirements and task parameters (such as data size) to the controller near the base station through LTE connection [4]. Cloud, fog and IoT nodes will send their own processing rates, task arrival rates to the controller. Based on these parameters and requirements of an application request, the controller will allocate the corresponding computing resource to the request. If controller assigns the task of a request to the fog or cloud, the IoT node will send the task data through LTE link to the base station, and the base station will transfer the data to fog or cloud through the wired backhaul [9]. Fog and cloud will return the processing results to the base station in the same way. The base station finally returns the processing results to the corresponding IoT node.

As shown in Fig. 1b, controller has a virtualization module that virtualizes IoT, fog and cloud nodes into a resource pool. It can achieve flexible scheduling by interacting with these nodes at different geographical locations, helping to realize efficient request processing by allocating appropriate resource to tasks. There is also an access control mechanism in controller ensuring that it can always assign tasks to the authenticated fog and cloud nodes. The basic module of controller is responsible for information communication. The resource allocators on the fog and cloud track the available resource of them, such as CPUs, etc, and allocate VMs to execute the assigned tasks of corresponding application request. They also map VMs to physical resource finding suitable VMs to handle the given tasks. The application cores in the infrastructure devices, fog and cloud have the application program which can handle the corresponding tasks. The energy analysis module of the IoT device determines whether the device can serve offloaded tasks of other devices according to the energy and time constraints. The program partitioning part is to divide the given application into several subtasks.

Fog layer can handle most of the tasks to reduce the overall service delay and the energy consumption of application request [8]. Thus, in this paper, the ETCORA method adopted by controller first considers whether a task is more suitable to be executed on the local IoT node or on the fog node according to the overheads on them. If the overhead on the fog node is greater than that of the IoT device, it will further consider whether the cloud node can be a better choice. If the overhead on cloud is smaller, the task will be assigned to cloud otherwise the local IoT node. The specific description of method ETCORA will be shown in Section 5.

3.2 Communication and computation models

This section first introduces the wireless access communication model in the IoT-fog-cloud environment and then the computation models on the three layers are described.

(1) Communication model The wireless base station can be the 3G or 4G macro-cell or small-cell base-station [20], which mainly manages the uploading and downloading transmission of IoT devices. Assuming that there are C wireless channels, denoted as $C = \{1, 2, ..., c\}$. $c_{n,m} \in \{0\} \cup C$ represents that the task m of request n is chosen to be offloaded to the fog or cloud through the wireless channel $c_{n,m} > 0$. $c_{n,m} = 0$ means that the task m will be executed on the local device. Let the channel allocation scheme for tasks of all requests be $S = (c_{1,1}, ...c_{1,m}, ..., c_{n,1}, ..., c_{n,m})$, if the task m of request n is assigned to the fog or cloud, then the data upload rate through wireless channel $c_{n,m}$ is shown as Eq. 1.

$$r_{n,m}(S) = \omega log_2(1+\frac{TP_{n,m}CN_{n,m}}{\varpi_0 + \sum\limits_{i\neq n, j\neq m, a_{i,j}=1} TP_{i,j}CN_{i,j}}) \quad (1)$$

Where ω is the bandwidth of the channel, $TP_{n,m}$ is the transmission power for the task m of request n (determined by the wireless base station according to the power control algorithm, such as [22] and [23]). $CN_{n,m}$ is the channel gain between device n and base station s for task m, and ϖ_0 is the background noise power.

Suppose the task m of request n has the size of input data (such as program code and input parameter) denoted by $b_{n,m}$, and the total number of CPU cycles required to execute this task denoted by $d_{n,m}$. It can only be executed on one of the places like the local IoT device, the fog and the cloud by offloading. So the next we will introduce the computation models on the local, fog and cloud.

(2) Local computation model Assume task m will be executed on the local device, $f^l_{n,m}$ represents the computing capability of device n (the number of CPU cycles per second). The local execution time and energy consumption for task m are shown as Eqs. 2 and 3.

$$t^l_{n,m} = \frac{d_{n,m}}{f^l_{n,m}} \quad (2)$$

$$E^l_{n,m} = \gamma_n d_{n,m} \quad (3)$$

Where γ_n is a coefficient that records the energy consumption per CPU cycle and can be obtained by [15]. Because for most applications (for example, face recognition), its computing outcome is smaller than its corresponding input data size (including mobile system settings, program code and input parameters), we do not consider the receiving time of the results from fog and cloud [14].

Before task m being executed, all of its predecessor tasks must be completely finished. Therefore, we will introduce the definition of prepared time of a task [10].

Definition 1 (prepared time) The prepared time of a task is defined as the earliest completing time for all the predecessor tasks of it. Therefore, for local computing, the prepared time $PT^l_{n,m}$ of task m from request n is shown by Eq. 4.

$$PT^l_{n,m} = \max_{k \in pare(m)} \{CT^l_{n,k}, CT^f_{n,k}, CT^c_{n,k}\} \tag{4}$$

Where $pare(m)$ is a set which contains all predecessor tasks of task m. Like [14], we do not consider the time of receiving outcome from fog and cloud, meaning that the task m can be executed immediately after the task k is completed. Therefore, the completion time $CT^l_{n,m}$ of the task m from request n on the local device is shown as Eq. 5.

$$CT^l_{n,m} = PT^l_{n,m} + t^l_{n,m} \tag{5}$$

According to Eqs. 3 and 5, the energy and time cost (ETC) of task m from request n can be obtained by Eq. 6.

$$K^l_{n,m} = \lambda^t_{n,m} CT^l_{n,m} + \lambda^e_{n,m} E^l_{n,m} \tag{6}$$

Where $\lambda^t_{n,m}, \lambda^e_{n,m} \in \{0, 1\}$ represent the weights for compuation time and energy consumption respectively. For flexibility, $\lambda^t_{n,m}, \lambda^e_{n,m} \in [0, 1]$ are also possible. In practice, the multi-attribute utility method can be used to determine the weights of computation time and energy consumption in the multi-standard decision making theory [25].

(3) Computation model on fog and cloud If a task is assigned to fog or cloud, the corresponding IoT node then will send the task data to fog or cloud through wireless access. According to the previous communication model, the requiring transmission time and energy consumption for task m of request n to be offloaded to the fog can be obtained by Eqs. 7 and 8. The transmission time $T^{c,trs}_{n,m}(S)$ and energy consumption $E^{c,trs}_{n,m}(S)$ of the case that task be offloaded to the cloud can be obtained in the same way. Where L_n is the tail energy due to the IoT device will continue to hold the channel for a while even after the data transmission [10]. Considering that the parameters such as $T^{c,trs}_{n,m}(S)$, $E^{c,trs}_{n,m}(S)$, $t^{c,exe}_{n,m}$, $PT^c_{n,m}$, $CT^c_{n,m}$, $K^c_{n,m}$ for the offloading to cloud case is similar to the offloading to fog case, for simplicity, we only introduce the parameters computation for the offloading to fog case.

$$T^{f,trs}_{n,m}(S) = \frac{b_{n,m}}{r^f_{n,m}(S)} \tag{7}$$

$$E^{f,trs}_{n,m}(S) = TP_{n,m} T^{f,trs}_{n,m}(S) + L_n \tag{8}$$

From Eqs. 7 and 8, we can get that lower data transmission rate $r_{n,m}$ will lead to longer transmission time and higher energy consumption. The execution time of task m from request n on fog is described by Eq. 9. Similarly, considering the dependencies of tasks, the prepared time of task m on fog is shown as Eq. 10.

$$t^{f,exe}_{n,m} = \frac{d_{n,m}}{f^f_{n,m}} \tag{9}$$

$$PT^f_{n,m} = max\{T^{f,trs}_{n,m}, \max_{k \in pare(m)} \{CT^f_{n,k}, CT^c_{n,k}\}\} \tag{10}$$

Where $\max_{k \in pare(m)} \{CT^f_{n,k}, CT^c_{n,k}\}$ denotes that all the predecessor tasks of task m that offloaded to fog and cloud have been executed. The formula explains that as long as the data of the task m has been offloaded to fog, or all of its corresponding previous tasks have been executed on fog and cloud, the task m can begin to execute immediately. Particularly, when the outcome receiving time of the task m is ignored, the completion time of task m on fog, $CT^f_{n,m}$, is the sum of the time of its prepared time and the execution time at fog. It is described by Eq. 11.

$$CT^f_{n,m} = t^{f,exe}_{n,m} + PT^f_{n,m} \tag{11}$$

According to Eqs. 8 and 11, the energy and time cost (ETC) for task m from request n on the fog is shown as Eq. 12.

$$K^f_{n,m} = \lambda^t_{n,m} CT^f_{n,m} + \lambda^e_{n,m} E^{f,trs}_{n,m}(S) \tag{12}$$

4 Problem formulation

For the request n with a task set M from an IoT device, the corresponding energy and time cost is described by Eq. 13.

$$K_n = \sum_{m=1}^{M} K_{n,m} = \sum_{m=1}^{M} a_{n,m} K^l_{n,m} + (1 - a_{n,m}) K^o_{n,m} \tag{13}$$

This paper is mainly to provide the optimal computation offloading strategy S^* and transmission power strategy P^*, so that the energy and time cost of application requests can be minimized. Where $K^o_{n,m}$ represents the energy and time cost of task m from request n on fog ($K^l_{n,m}$) or cloud ($K^c_{n,m}$). Therefore, the task allocation problem of all devices can be considered as the following constraint minimization problem:

$$\begin{aligned} &\min_{S,P} \sum_{n=1}^{N} K_n && (14)\\ &s.t : \forall m \in M, \forall n \in N \\ &\quad C1 : \sum\nolimits_{m=1}^{M} a_{n,m} CT^{l}_{n,m} + (1-a_{n,m}) CT^{o}_{n,m} \le T_{n,max} \\ &\quad C2 : \sum\nolimits_{m=1}^{M} a_{n,k} CT^{l}_{n,k} + (1-a_{n,k}) CT^{o}_{n,m} \le PT^{l}_{n,m} \\ &\quad C3 : CT^{t}_{n,m} \le PT^{o}_{n,m} \\ &\quad C4 : \max_{k \in pare(m)} CT^{o}_{n,m} \le PT^{o}_{n,m} \\ &\quad C5 : a_{n,m} \in \{0,1\} \end{aligned}$$

Where $k \in pare(m)$, $S = \{c_{n,m} | n \in N, m, \in M\}$, $P = \{TP_{n,m} | n \in N, m \in M\}$. $C1$ is the completion time constraint, indicating that the total completion time of all tasks of an application request n should meet the deadline $T_{n,max}$ constraint. $C2$ ensures that a task assigned to the local device can start to execute only if all predecessor tasks of it have finished completely. $C3$ and $C4$ show that a task offloaded to the fog or cloud can start executing only if the task data has been transmitted to the fog or cloud, or all the predecessor tasks of it on the fog or cloud have totally completed. $C5$ denotes that a task only can be executed on one of the places: the local device, fog, or the cloud.

The main challenge to solve the problem in Eq. 14 is the integer constraint $a_{n,m} \in \{0,1\}$. It makes the problem in Eq. 14 become a mixed integer programming problem which is generally non-convex and NP-hard. Thus, similar to [26, 27], we first relax the variable $a_{n,m}$ to be the real number from 0 to 1. As $0 \le a_{n,m} \le 1$, according to [31], the objective function in Eq. 14 with constraints $C1 - C5$ is jointly convex with respect to the optimization variables $a_{n,m}$, $TP_{n,m}$. That is, the optimization problem in Eq. 14 has a zero duality gap and satisfies the Slater's qualification. Because the strong duality, any pair of primal and dual optimal points must satisfy the KKT conditions [31]. That is, the optimal solution of the primal problem in Eq. 14 can be derived from it's dual problem.

Thus, let the Lagrange function of the problem (14) be $L(\Psi, \mu, S, P)$. The Lagrange multiplier $\Psi = [\Psi_n, n = 1, 2, ..., N]^T$ satisfies the completion time constraints $C1$, $C2$. Ψ_n denotes the total completion time of request n would not exceed the required maximum completion time. The Lagrange multiplier $\mu = [\mu_{n,m}, m = 1, 2, ..., M; n = 1, 2, ..., N]^T$ represents the corresponding constraints $C3$, $C4$ for the task dependency requirement on the tasks that are not executed by the local devices, indicating that the task m can start to be executed on fog or cloud only when the required data transmission has finished completely. Thus, the dual problem of Eq. 14 can be expressed as Eq. 15.

$$\max_{\Psi,\mu} \min_{S,P} L(\Psi, \mu, S, P) \qquad (15)$$

The dual problem (15) can be divided into two layers: the internal minimization function and the external maximization function. The former is composed of N sub-problems with similar structure and can be solved in a distributed way. The latter is the main problem. So, next we will introduce the algorithm to solve the two-layer problem including computation offloading selection and transmission power allocation.

5 Energy and time efficient computation offloading and resource allocation

This section introduces the ETCORA algorithm, which is mainly adopted by controllers in fog layer to assign appropriate resource to tasks. It consists of two parts: computation offloading selection and transmission power allocation.

5.1 Computation offloading selection

The computation offloading selection is mainly to determine which task of the request needs to be offloaded to the fog or cloud, satisfying the task dependence and the number of tasks executed on fog layer as more as possible, making the energy and time cost of each application request as less as possible. For a task, firstly comparing the local execution overhead of it with the execution overhead on the fog node. If the local execution is larger, the task will be offloaded to the fog layer. otherwise, further considering whether it is possible to be offloaded to the cloud layer. If the execution overhead of the task on the cloud layer is smaller, it will be eventually offloaded to the cloud, otherwise it will stay on the local device.

Let $OH^{l}_{n,m} = K^{l}_{n,m} + \Psi_n CT^{t}_{n,m}$ be the overhead of task m from request n. $OH^{f}_{n,m} = K^{f}_{n,m} + \Psi_n CT^{f}_{n,m}$ be the overhead of task m on fog ($OH^{c}_{n,m}$ be the overhead of it on cloud), then the optimal computation offloading selection strategy can be obtained by solving the optimization problem (16) satisfying with $C2$, $C4$, $C5$.

$$\min_{a_{n,m}} OH^{l}_{n,m} + a_{n,m}(OH^{f}_{n,m} - OH^{l}_{n,m}) \qquad (16)$$

Obviously, Eq. 16 is a linear function of $a_{n,m}$. If $OH^{f}_{n,m} \ge OH^{l}_{n,m}$ (for the case $OH^{c}_{n,m} \ge OH^{l}_{n,m}$ is the same), then Function (16) can obtain the minimum value when $a_{n,m} \in [0,1]$ gets the minimum value. If $OH^f_{n,m} < OH^l_{n,m}$, Eq. 15 can get the maximum value when $a_{n,m}$ reaches the minimum. Thus, the computation offloading selection strategy can be obtained as Eq. 17.

$$a_{n,m} = \begin{cases} 0, otherwise \\ 1, OH^f_{n,m} < OH^l_{n,m} (OH^c_{n,m} < OH^l_{n,m}) \end{cases} \tag{17}$$

As for the comparing between $OH^l_{n,m}$ and $OH^c_{n,m}$ is the same. The whole process of offloading selection is shown as Algorithm 1.

5.2 Transmission power allocation

The transmission power allocation is mainly to allocate transmission power to the tasks of each request optimally, minimizing the energy and time cost of task executed on fog or cloud. Therefore, the strategy is applied to tasks that need to be offloaded to fog or cloud, which can be obtained by solving the optimization problem shown as Eq. 18 satisfying with $C2$, $C4$. As for the case that a task needs to be offloaded to cloud is the same, we just introduce the solution of the transmission power strategy for tasks that would be offloaded to the fog.

$$Y(TP_{n,m}) = \min_{TP_{n,m}} \sum_{n=1}^{N}\sum_{m=1}^{M} K^f_{n,m} + \Psi_n CT^f_{n,m} + \mu_{n,m}(CT^t_{n,m} - PT^f_{n,m}) \tag{18}$$

According to Eqs. 7–12, there are two cases for $PT^f_{n,m}$, correspondingly, there are two cases for $Y(TP_{n,m})$.

Case 1 $CT^t_{n,m} > \max\limits_{k \in pare(m)} CT^f_{n,m}$.

If $PT^f_{n,m} = CT^t_{n,m}$, then:

$$Y(TP_{n,m}) = \sum_{n=1}^{N}\sum_{m=1}^{M} (\lambda^t_{n,m} + \Psi_n)[t^{f,exe}_{n,m} + CT^t_{n,m} + \lambda^e_{n,m} E^{f,trs}_{n,m}(S)]x \tag{19}$$

Where $CT^t_{n,m} = t^{f,exe}_{n,m} + PT^{trs}_{n,m}$. $PT^{trs}_{n,m}$ represents the time that task m is ready to be transmitted to fog or cloud, in order to ensure the task dependency relationship requirement, it is a constant for task m.

Case 2 $CT^t_{n,m} \le \max\limits_{k \in pare(m)} CT^f_{n,k}$.

Algorithm 1 Computation offloading selection.

```
Input: Task $m$, $pare(m)$: parent tasks of $m$,
       $b_{n,m}$, $d_{n,m}$, $\lambda^e_{n,m}$, $\lambda^t_{n,m}$, $\psi_n$ and
       $\mu_{n,m}$, $f_{n,m}$, $TP_{n,m}$ and iteration index $t$;
Output: $a_{n,m}$
1  Offloading-selection()
2  Compute $r_{n,m}$, $CT^l_{n,m}$, $E^l_{n,m}$ by Equations (1)-(3);
3  if $pare(m)$ is empty then
4      $PT^l_{n,m} = 0$, $T^{f,trs}_{n,m} = 0$;
5      else
6          Get $PT^l_{n,m}$, $PT^f_{n,m}$ by Equation (4), (10);
7      end
8  end
9  Compute $CT^l_{n,m}$, $K^l_{n,m}$ by Equations (5)(6);
10 Compute $OH^l_{n,m} = K^l_{n,m} + \psi_{n,m} CT^l_{n,m}$;
11 Get $T^{f,trs}_{n,m}$, $E^{f,trs}_{n,m}$, $t^{f,exe}_{n,m}$ by Equations (7)-(9);
12 Get $CT^f_{n,m}$ by Equation (11);
13 (For the case of comparing between local mode and
   cloud mode, it is the $T^{c,trs}_{n,m}$, $E^{c,trs}_{n,m}$, $t^{c,exe}_{n,m}$, $CT^c_{n,m}$ here
   that need to be obtained, the following is similar);
14 if $pare(m)$ is empty then
15     $PT^f_{n,m} = T^{f,trs}_{n,m}$;
16     else
17         Compute $PT^f_{n,m}$ by Equation (11);
18     end
19 end
20 Get $CT^f_{n,m}$, $K^f_{n,m}$ by Equations (11) and (12);
21 Get $OH^f_{n,m} = K^f_{n,m} + \psi_{n,m} CT^f_{n,m}$;
22 if $OH^f_{n,m} < OH^l_{n,m}$ then
23     $a_{n,m} = 1$;
24     else
25         $a_{n,m} = 0$;
26     end
27 end
```

For example, $PT^f_{n,m} = \max\limits_{k \in pare(m)} CT^t_{n,m}$, which is independent of $TP_{n,m}$, then, $Y(TP_{n,m})$ is shown as Eq. 20.

$$Y(TP_{n,m}) = \sum_{n=1}^{N}\sum_{m=1}^{M} (\lambda^t_{n,m} + \Psi_n)[t^{f,exe}_{n,m} + PT^f_{n,m} + \lambda^e_{n,m} E^{f,trs}_{n,m}(S) + \mu_{n,m}(CT^t_{n,m} - PT^f_{n,m})] \tag{20}$$

Based on the above mentioned, the transmission power allocation strategy in Case 1 then can be obtained. From Theorem 1 [10], $Y(TP_{n,m})$ in Eq. 20 can converge on $TP_{n,m}$, so by using the KKT condition [31], the transmission power allocation strategy is satisfied with Eq. 21.

$$\frac{\delta_{n,m}}{\varpi_{n,m}+TP_{n,m}CN_{n,m}}+1=ln(1+\frac{TP_{n,m}CN_{n,m}}{\varpi_{n,m}}) \quad (21)$$

Where $\delta_{n,m}=(\lambda^{t}_{n,m}+\Psi_n)\frac{CN_{n,m}}{\lambda^{e}_{n,m}}-\varpi_{n,m}$ and $\varpi_{n,m}=\varpi_0+\sum_{i\neq n,j\neq m,a_{i,j}=1}TP_{i,j}CN_{i,j}$ indicates the noise power and interference when the access point s receives the data of task m which belongs to request n from the transmitter.

Equation 21 can be divided into two equations as $\varphi(TP_{n,m})=(\frac{\delta_{n,m}}{\varpi_{n,m}+TP_{n,m}CN_{n,m}}+1)$ and $\phi(TP_{n,m})=ln(1+\frac{TP_{n,m}CN_{n,m}}{\varpi_{n,m}})$. Equation 21 is a transcendental equation, usually there is not a closed form solution for $TP_{n,m}$. We can get an approximate solution by Newton iteration method, for example, the transmission power can be updated iteratively by Eq. 22.

$$TP_{n,m}(t+1)=TP_{n,m}(t)-\frac{\varphi(TP_{n,m}(t))-\phi(TP_{n,m}(t))}{\varphi'(TP_{n,m}(t))-\phi'(TP_{n,m}(t))} \quad (22)$$

Where $\varphi'(\cdot)$ and $\phi'(\cdot)$ represent the first derivative of the two functions on $TP_{n,m}(t)$. Because $\varphi'(TP_{n,m}(t))\neq\phi'(TP_{n,m}(t))$ and the initial value of $TP^{\star}_{n,m}$ is close to 0, according to [10], the Newton iteration method is convergent in this case.

Similar to the Case 1, the $TP_{n,m}$ in Case 2 can be obtained by Eq. 23.

$$TP_{n,m}(t+1)=TP_{n,m}(t)-\frac{\chi(TP_{n,m}(t))-\phi(TP_{n,m}(t))}{\chi'(TP_{n,m}(t))-\phi'(TP_{n,m}(t))} \quad (23)$$

Where $\chi(TP_{n,m}(t))=\frac{\beta_{n,m}}{\varpi_{n,m}+TP_{n,m}CN_{n,m}}$, $\beta_{n,m}=\frac{\mu_{n,m}CN_{n,m}}{\lambda^{e}_{n,m}-\varpi_{n,m}}$.

For the the external maximization function of problem 15, the sub-gradient method can be adopted to solve it. Therefore, the Lagrange multiplier of certain S, P can be updated according to Eqs. 24 and 25.

$$\Psi_n(k+1)=[\Psi_n(k)+\vartheta(k)(T_{n,max}-\sum_{m=1}^{M}(1-a_{n,m})CT^{l}_{n,m}+a_{n,m}CT^{o}_{n,m})]^{+} \quad (24)$$

$$\mu^{k+1}_{n,m}=[\mu_{n,m}(k)+\vartheta(k)(PT^{o}_{n,m}-CT^{o}_{n,m})]^{+} \quad (25)$$

Where $k>0$ is the number of iteration, ϑ is the positive iterative step. The updated Lagrange multiplier in Eqs. 24 and 25 can be used to update the computation offloading selection strategy and the transmission power allocation strategies.

Algorithm 2 Energy efficient task offloading and resource allocation algorithm.

```
Input: $M$: a sequence of tasks of an application request
       from device $n$, $Iter_{max}$: maximum number of
       iterations, $\{pare(m)\}$: a set of predecessor tasks
       with task $m \in M$, $\theta$: an infinitesimal number;
Output: {S,P}: Optimal resource allocation policy;
1  Initialize $b_{n,m}$, $d_{n,m}$, $\lambda^{e}_{n,m}$, $\lambda^{t}_{n,m}$, $\psi_{n,m}$ and
   $\mu_{n,m}$, $\{f_{n,m}\}$, $\{TP_{n,m}\}$ and iteration index $t \leftarrow 1$,
   flag=0;
2  for $m = 1$ to $M$ do
3      while $t \leq Iter_{max}$ and
       $|\mu_{n,m}(t+1)-\mu_{n,m}(t)| > \theta$ do
4          /* Whether to fog */
5          Get $a_{n,m}$ by Computation offloading selection
           algorithm;
6          if ($a_{n,m} == 1$) then
7              Offload $m$ to fog;
8              flag=1;
9              /* Transmission power allocation */
10             if ($T^{f,trs}_{n,m} > \max_{k \in pare(m)} \{CT^{f}_{n,k}\}$) then
11                 Compute $TP_{n,m}$ by Equation (22)
                   using Newton iteration method;
12             else
13                 Compute $TP_{n,m}$ by Equation (23)
                   using Newton iteration method;
14             end
15         end
16         /* Not to fog, further considering the
           cloud*/
17         else
18             Get $a_{n,m}$ by Computation offloading
               selection algorithm;
19         end
20     end
21     if ($a_{n,m} == 1$ and $flag == 0$) then
22         Offload $m$ to cloud;
23         flag=1;
24         Executing transmission power allocation
           the same as the step 10 to the step 15;
25     else
26         Task $m$ will be assigned to the current
           IoT device;
27     end
28     end
29     Update $\psi_{n,m}(t+1)$, $\mu_{n,m}(t+1)$ by
       Equations(24) and (25);
30     $t = t + 1$;
31     flag=0;
32   end
33 end
```

5.3 The enforcement of ETCORA algorithm

On the whole, the ETCORA is shown by Algorithm 2. The input parameters include a set of tasks of request n, M; the maximum number of iterations, $Iter_{max}$; the parent task set $\{pare(m)\}$ and an infinitesimal number θ. The output is the optimal resource allocation policy.

Fog layer can handle most of the tasks to reduce the overall service delay and the energy consumption of application requests [8]. Thus, for the purpose of having as more tasks to be executed on the fog as possible, the first step of ETCORA method adopted by controller is to consider whether a task is more suitable to be executed on the local IoT node or on the fog according to the overheads on them. If the overhead on the fog is greater than that of the IoT device, it will further consider whether the cloud can be a better choice. If the overhead on cloud is smaller, the task will be assigned to cloud otherwise the local IoT node.

From line 4 to line 15 of Algorithm 2, it is to judge whether the current task needs to be offloaded to fog. That is, the computation offloading selection in Algorithm 1 will be invoked here to choose a better executing place (local/fog). If $a_{n,m} = 1$, meaning it is beneficial to offload the current task to fog, then the dynamic transmission power allocation needs to be done to allocate appropriate transmission power to the task (Line 10-14). If $a_{n,m} = 0$, indicating that it is better for the task to be executed on the local device compared with the execution on fog, but we will further consider whether the cloud can be better than the local execution case. In other words, the computation offloading selection in Algorithm 1 will be invoked again here to make a decision between local and cloud. If $a_{n,m} = 1$, the task will be offloaded to cloud and the transmission power allocation also will be invoked, else the task will be eventually assigned to the local device (Line 16-28). In the next, we are going to introduce the simulation experiments conducted to show the effectiveness of the proposed method.

6 Performance simulation

Due to lack of the realistic testbed, we evaluate the effectiveness of the proposed method by iFogSim [25] to simulate an IoT-fog-cloud environment (including 500 IoT nodes, 60 fog nodes, and 20 cloud server nodes). iFogSim is an extension to the CloudSim framework [26], which has been widely used to simulate different computing modes. We extend iFogSim to obtain the IoT-fog-cloud architecture in Section 3. We modify the Application and AppModule classes to contain the sub-tasks and task relationships of the application, and the corresponding completion time constraint. FogDevice is extended to FogServer to represent fog nodes and controllers. The definition of PlacementOptimization interface facilitates the implementation of various resource allocation methods and the unified access to simulation environment to analyze and execute the configured results.

According to [10, 32], the size of task $b_{n,m}$ and the number of CPU cycles $d_{n,m}$ required by corresponding task follow the Gauss distribution $GN(\mu_1, \sigma_1^2)$ and $GN(\mu_2, \sigma_2^2)$ respectively, where $\mu_1 = 200KB$ and $\mu_2 = 1000$ Mega circles, $\sigma_1 = 50$, $\sigma_2 = 100$. It is assumed that the application requests generated by a IoT device in a certain domain satisfy the Poisson distribution, and the generating rate is γ_i, meaning the request generating rate is γ in domain i.

We divide the face recognition in [33] into 100 tasks using computational segmentation method in [34]. The directed acyclic graph shown in Fig. 2 describes the task relationships of two sub-parts of the application. The nodes in the graph represent different tasks, and the nodes with unidirectional arrow represent the successively dependent relationship between the tasks. Only when the direct previous tasks are completed, the corresponding task can begin to execute.

According to the communication model in Section 3.2, the bandwidth between IoT devices and fog is a random value in [5, 10] MHz, the bandwidth between fog and cloud is a random value in [10, 25] MHz, the bandwidth between IoT devices and cloud is a random value in [25, 50]MHz [20]. The background noise power ϖ_0 is 50dBm [10, 35], the channel gain $CN_{n,m} = ds_{n,m}^{-\xi}|h_{n,m}|^2$, where $ds_{n,m}$ is the Euclidean distance between the IoT device n and the access point s, and $\xi = 4$ represents the path loss factor [5, 10]. $h_{n,m}$ is the corresponding Rayleigh fading channel coefficient that follows the distribution of $N(0, 1)$ [36].

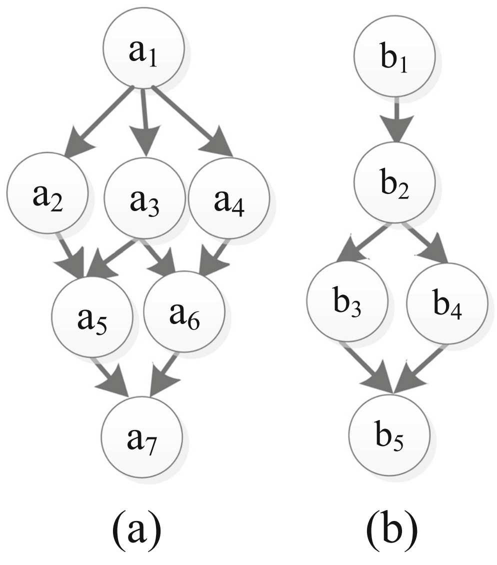


Fig. 2 Part of sub-tasks dependency

We set the initial decision weight $\lambda^t_{n,m}$and $\lambda^e_{n,m}$ to be 0.5, 0.5. According to [25], it is assumed that the processing capability of the IoT device is a random value in [0.5, 1.0] GHz, and the processing capability of the fog node is a random value in [5, 10] GHz, and the processing ability of the cloud node is a random value in [50, 100] GHz.

According to Section 2, the methods DMP [5] and ADMMD [8] are applied to the similar scenario as this paper, and they have some kind of similar goal of optimization with the work of this paper, so we consider them as the baselines in the simulation experiments. To evaluate the performance of the proposed method, the following four questions are concerned.

RQ1: A system architecture with three layers (IoT, fog, cloud) is presented in this paper, aiming to take full advantages of fog and cloud. Thus, we evaluate the average ETCs of application requests under four different modes to show the difference among them.

RQ2: We embed the Newton iteration method into ETCORA to solve the near optimal transmission power allocation problem for task which is not executed on the local node. So we evaluate the convergence of it, the influence of weights $\lambda^e_{n,m}$, $\lambda^t_{n,m}$ on the energy consumption and computation time of tasks which have different number of parent tasks.

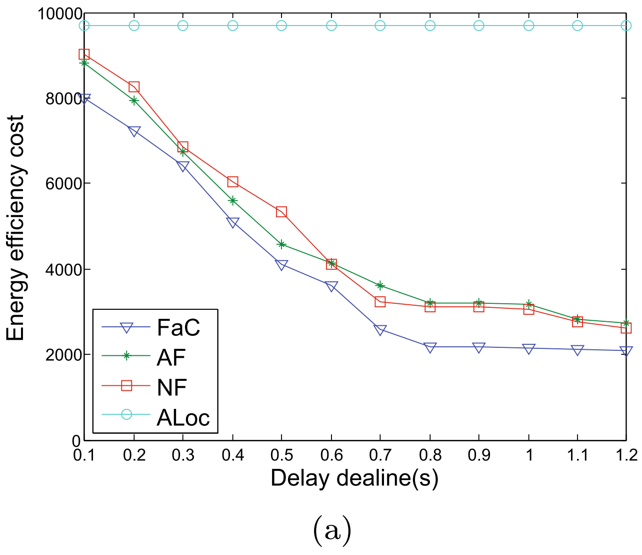
RQ3: This paper is to improve the service delay and the energy consumption required by the request processing. Each application request has a set of tasks under certain task dependencies, the effect of task deadline on ETC of the request is evaluated for DMP [5], ADMMD [8] and the ETCORA. Also, the average energy consumption and computation time of the application request are compared when there are different number of application requests for DMP [5], ADMMD [8] and the ETCORA.

RQ4: On the basis of RQ3, the effect of different sizes of input data for tasks on the energy consumption and computation time of the request is evaluated for DMP [5], ADMMD [8] and the ETCORA.

6.1 Cooperation of fog and cloud

For RQ1, we compare the average ETCs under 4 different modes: the all local execution mode (ALoc), no fog (NF) mode, all fog (AF) mode and the fog and cloud (FaC) mode. ALoc mode means all tasks will be executed on the local IoT node. NF mode means IoT node will process the tasks of a request itself or send tasks to the cloud. AF mode means IoT node will process the tasks of a request itself or offload them to the fog. FaC mode means IoT node will process the tasks of a request itself or send them to fog or the cloud.

Figure 3a shows the ETCs of the application request in four different modes. From the graph, we can see that



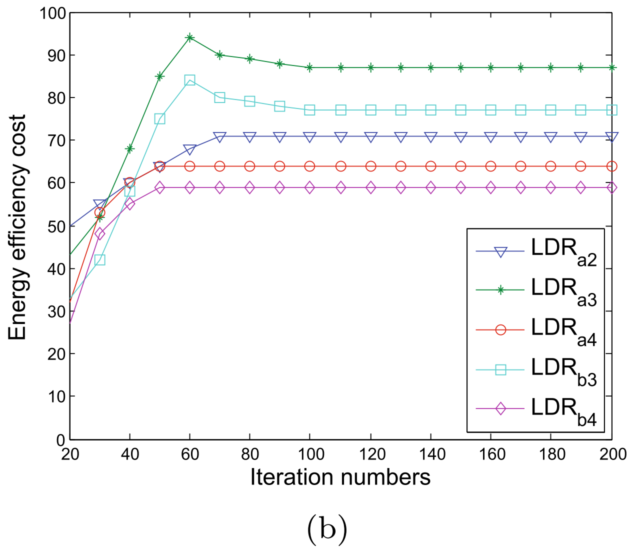
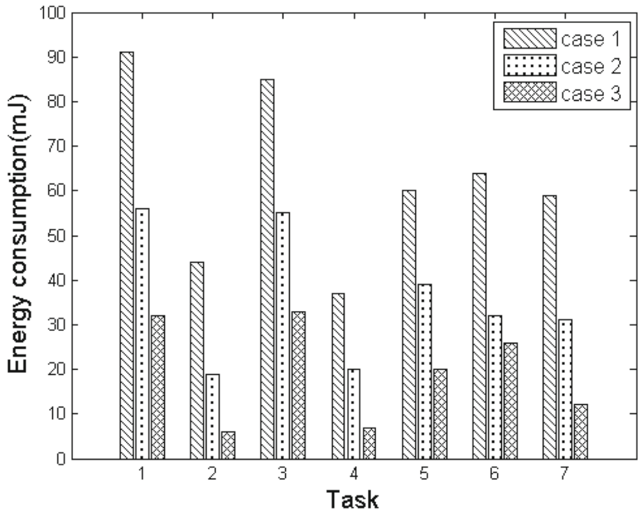


Fig. 3 Cooperation of fog and cloud (**a**), Convergence with different task complexity (**b**)

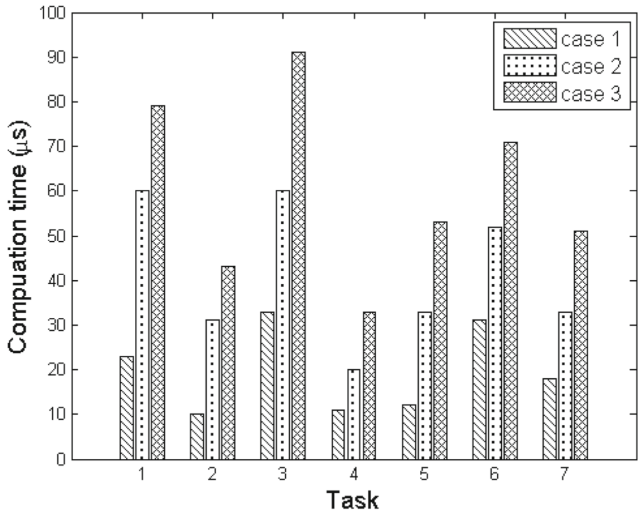
the ETCs in ALoc mode almost have no change with the relaxation of deadline. Because in ALoc mode, all tasks are executed on the local IoT node, it is hard to obtain greater optimization on the energy and time cost although the deadline constraint is more loose. The ETCs in NF mode and AF mode are very similar, while the ETCs in FaC mode are obviously better than the other three modes. This is because the FaC mode contains both advantages of AF and NF compared with AF and NF, and the use of resource is more flexible. And with the increase of the deadline, the ETCs decreasing in NF, AF and FaC modes tend to be slow. This is because with the relaxation of the deadline constraints, the allocation of resource under the 3 modes are more free, meaning the constraints of the use of resources are not too strict. Therefore, the corresponding optimization of energy and time cost will reach a relatively fixed value.

6.2 Convergence and impact of weight

According to RQ2, in this part, we evaluate the convergence of the proposed method and the effect of task complexity on ETC. Figure 3b shows the ETCs of tasks $a_2, a_3, a_4(b_3, b_4)$ and the convergence of the proposed method. According to [10], the load data rate $LDR_{n,m} = b_{n,m}/d_{n,m}$ represents the complexity of tasks. For tasks $a_2, a_3, a_4(b_3, b_4)$, for example, the complexity of them are 9.522, 11.35, 4.742 (11.056, 3.011) respectively(we choose tasks a_2, a_3, $a_4(b_3, b_4)$ because they share the same predecessor tasks). It can be seen from the graph that the proposed method can make all tasks converge at no more than 100 iterations.



(a)



(b)

Fig. 4 Effect of weight on energy consumption (**a**) and computation time (**b**)

Among them, the higher the task complexity is, the slower the convergence speed is, and the higher corresponding ETC is. This phenomenon shows that in the process of computing task segmentation, a reasonable task complexity set has a positive effect on reducing energy consumption and computation delay.

Figure 4 is a comparison of the energy consumption and computation delay of tasks $a_1, a_2, a_3, a_4, a_5, a_6, a_7$ under different combinations of $\lambda^e_{n,m}, \lambda^t_{n,m}$. We set three different cases for the combination of $\lambda^e_{n,m}$ and $\lambda^t_{n,m}$ [10]. For case 1, the $\lambda^e_{n,m}$ and $\lambda^t_{n,m}$ are set to be 0.2 and 0.8 respectively. For case 2, the $\lambda^e_{n,m}$ and $\lambda^t_{n,m}$ are both set as 0.5. For case 3, the $\lambda^e_{n,m}$ and $\lambda^t_{n,m}$ are set to be 0.8 and 0.2 respectively. As can be seen from the diagram, for a given task, with the decrease of the $\lambda^e_{n,m}$ value, the energy consumption of the task will increase, but the computation delay is on the contrary. The reason is that the smaller the $\lambda^e_{n,m}$ value, the lower the importance the energy consumption got during the allocation process (but the higher importance of the computation delay, so the shorter computation time spent), the $\Delta_{n,m}$ and $\phi(TP_{n,m})$are increasing, but the $TP_{n,m}$ is decreasing. That is, the transmission power for task executed in the fog or cloud will be declining, the corresponding transmission time will be higher, meaning the energy consumption will increase too.

6.3 Effect of task deadlines on ETC

Section 6.3 and 6.4 are presented to answer the RQ3. This subsection introduces the difference among DMP [5], ADMMD [8] and ETCORA with the indicator of ETC.

Figure 5 shows the ETCs corresponding to tasks with different completion time constraints in the application request. As can be seen from the diagram, the corresponding ETCs with relatively loose time constraint is smaller,

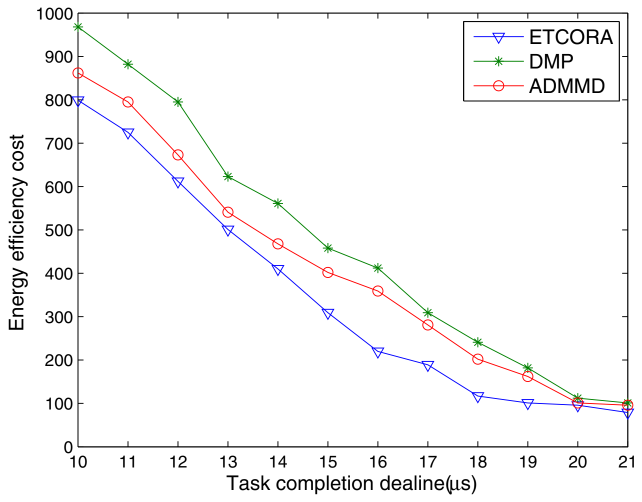


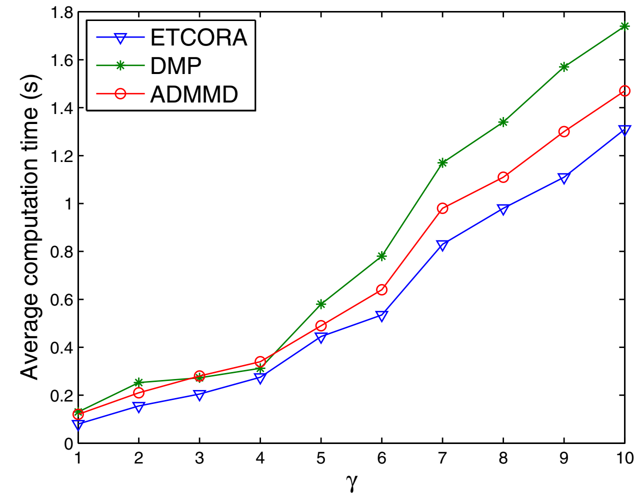
Fig. 5 Effect of task completion deadline

ETCORA is more obviously better than the other two methods. This is because, under the same condition, in ETCORA, tasks of application request will be assigned to the local/fog/cloud node according to the general overhead on the local/fog/cloud node, meanwhile the assignment of the task data transmission power is optimized for the fog execution or the cloud execution. The graph shows that when the time constraints are magnified to a certain extent, the energy and time cost of the 3 methods tend to be similar. This maybe because the looser constraints of the task requirements and the relative adequacy of the required resource, the more similar task assignments of the 3 methods. For example, a task is allocated to the same place like the local, the fog or the cloud under these three methods, so the energy consumption costs of the methods could be very similar.

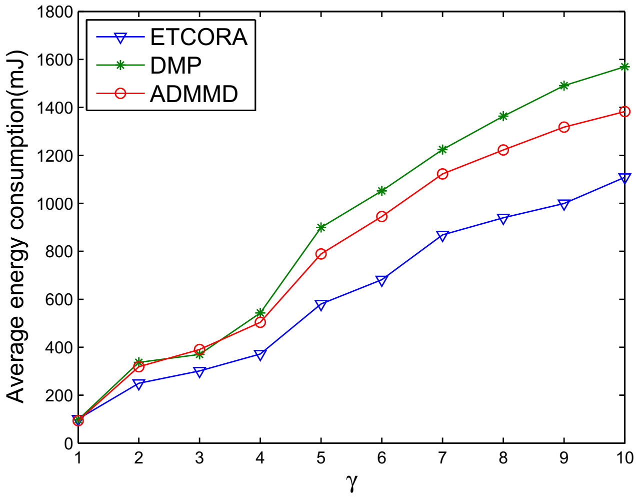
6.4 Impact of changing value of γ

In this section, the value of γ is set to be from 1-10, the average energy consumption and computation time of the application request is recorded. Different values of γ indicate different number of application requests at each time period. The greater the value of γ, the more application requests need to be processed accordingly. Figure 6 shows the average energy consumption and computation time corresponding to different values of γ. It can be seen from the diagram the average energy consumption and computation time of the 3 methods increase in an irregular trend. When the γ is less than 4, the corresponding growth trend is slow. This is because when the value is less than 4, the number of application requests is not very large, but when the value is greater than 4, the amount of application requests is obviously more than the former, so the more average energy consumption and the computation time are required. And the rate of increase is more obvious than that of the former.

Moreover, we can see from the graph that the average energy consumption and computation time of ADMMD and DMP are higher than that of ETCORA. This is because, on one hand, when the request number is increasing, the advantages of ETCORA using fog and cloud will be more obvious. On the other hand, As for a task of a request, ETCORA performs task allocation by selecting the local/fog/cloud that has the smallest energy cost under the task completion time constraint. If a task is not assigned to the local device, indicating that it should be offloaded to the fog or the cloud to be executed, then the most appropriate transmission power is assigned to the task for data transmission, so, ETCORA has a better performance compared to the other two algorithms. Interestingly, the average energy consumption of the 3 methods are similar when γ is 1 as shown in Fig. 6a. This maybe because the tasks in the current environment are less and the resources are comparatively abundant. Therefore, the assignment results of the 3 methods are very similar, and the corresponding energy consumptions are very similar. But with the increase of the number of requests, this phenomenon disappears.



(a)



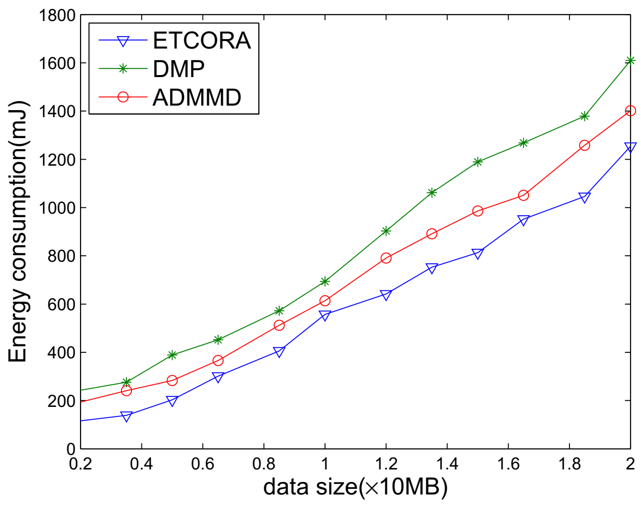
(b)

Fig. 6 Effect of changing value of γ on computation time (**a**) and energy consumption (**b**)

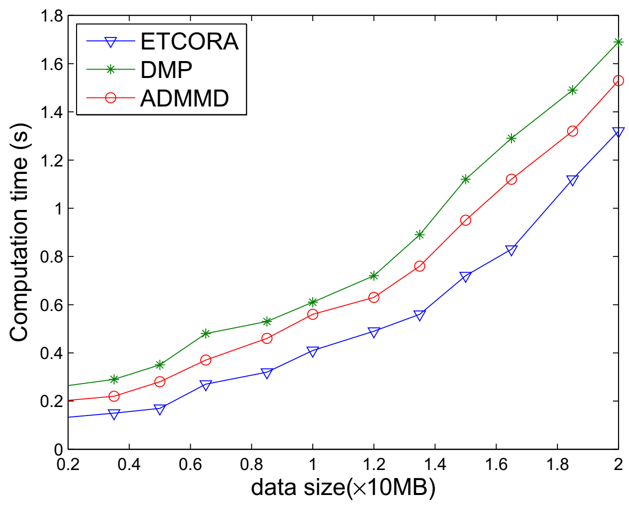
6.5 Effect of input data size

For RQ4, in this part, we compare the difference of the 3 methods in energy consumption and computation time when the size of input data is constantly changing.

Figure 7 shows the average energy consumption and computation time of the 3 algorithms. It can be seen from the diagram that when the γ value is certain, that is,



(a)



(b)

Fig. 7 Effect of input data size on energy consumption (**a**) and computation time (**b**)

the number of application requests in the corresponding time period is relatively certain, with the increase of the input data size of the task, the corresponding energy consumptions of ADMMD and ETCORA are less than DMP. Because both the former two are optimized for energy consumption. In addition, ETCORA takes into account the dependency of the tasks in the application, the advantages of fog and cloud, so that the corresponding energy consumption value is the lowest. In Fig. 7b, the corresponding computation times of the 3 algorithms increase with the increase of data size, which is due to the larger the size of the data, the higher the transferring and execution time required in the offloading and executing processes. Relatively, the overall computation time of ETCORA is lower than those of the other 2 methods. On one hand, because ETCORA takes the overall overhead of the local/fog/cloud into consideration when the task is assigned, it chooses the most suitable task to carry out the migration, which is to choose the most appropriate execution place, the local/fog/cloud, to execute the task. On the other hand, because ETCORA has a dynamic distribution of transmission power, it is possible to allocate the appropriate transmission power to each task and make full use of the energy consumption of the IoT device.

7 Conclusion and future work

This paper mainly deals with the research on the energy and time efficient computation offloading and resource allocation for IoT applications, which is aimed at reducing the energy consumption and completion time of IoT application requests, improving the computing power of IoT devices. We first proposes a general IoT-fog-cloud computing architecture that fully exploits the advantages of fog and cloud. Then, the energy and time efficient computation offloading and resource allocation is formulated into the energy and time cost minimization problem. We then propose the ETCORA algorithm including 2 parts: computation offloading selection and transmission power allocation to solve the problem. Finally, extensive simulations are carried out to verify the effectiveness of the proposed framework and algorithm compared with other two methods. Results show that our method indeed outperforms the other two methods in reducing energy consumption and completion time of requests.

Due to the limited condition, we do the simulation experiments to evaluate the effectiveness of the proposed method compared with the other two baseline methods. In the future, we are going to realize the proposed method in a realistic IoT-fog-cloud environment to extend the practicability of it. If so, it will benefit many aspects of our daily life such as the transportation, medical, industrial automation, smart home and emergency response. In addition, we will further consider the availability, utilization or/and cost of resources. As for the continuous tasks per second, the corresponding computing resources on the fog layer are precious, how to make full use of them are worth studying. Moreover, the relatively higher mobility of fog nodes compared with cloud resources is another factor need to be taken into consideration. Another two significant factors are the security and reliability of services, because they would also have direct impact on the performance of the IoT applications.

Acknowledgments This work is partially supported by the NSF of China under grants No. 61772200, and 61702334, Shanghai Pujiang Talent Program under grants No. 17PJ1401900. Shanghai Municipal Natural Science Foundation under Grants No. 17ZR1406900 and 17ZR1429700. Educational Research Fund of ECUST under Grant No. ZH1726108. The Collaborative Innovation Foundation of Shanghai Institute of Technology under Grants No. XTCX2016-20. The Humanities and Social Science Research Planning Fund of the Education Ministry of China under grant No.15YJCZH201.

References

1. Lu T, Chang S, Li W (2018) Fog computing enabling geographic routing for urban area vehicular network [J]. Peer Peer Netw Appl 11(4):749–755
2. Alansari Z, Soomro S, Belgaum MR et al (2018) The rise of internet of things (IoT) in Big Healthcare Data: Review and Open Research Issues [C]// International Conference on Advance Computing and Intelligent Engineering
3. Ding K, Jiang P (2018) RFID-based Production Data Analysis in an IoT-enabled Smart Job-shop [J]. IEEE/CAA J Autom Sin PP(1):1–11
4. Li H, Ota K, Dong M (2018) Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing [J]. IEEE Netw 32(1):96–101
5. Yousefpour A, Ishigaki G, Jue JP (2017) Fog computing: Towards minimizing delay in the internet of things [C]//. IEEE International Conference on Edge Computing. IEEE, pp 17–24
6. You C, Huang K, Chae H et al (2017) Energy-Efficient Resource allocation for Mobile-Edge computation offloading [J]. IEEE Trans Wirel Commun 16(3):1397–1411
7. Liu Y, Fan C, Liu H et al (2017) Cross-layer Cooperative Multichannel Medium Access for Internet of Things [J]. Peer-to-Peer Netw Appl 11(1):1–14
8. Chang Z, Zhou Z, Ristaniemi T et al (2018) Energy efficient optimization for computation offloading in fog computing system [C]// GLOBECOM 2017 - 2017. IEEE Global Communications Conference. IEEE, pp 1–6
9. Hussain MM, Alam MS, Beg MMS (2018) Fog Computing in IoT Aided Smart Grid Transition- Requirements, Prospects, Status Quos and Challenges [J]
10. Guo S, Xiao B, Yang Y et al (2016) Energy-efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing [C]//. IEEE International Conference on Computer Communications. IEEE INFOCOM, pp 1–9
11. Han T, Ansari N (2017) Network Utility Aware Traffic Load Balancing in Backhaul-Constrained Cache-Enabled Small Cell Networks with Hybrid Power Supplies [J]. IEEE Trans Mob Comput PP(99):1–1
12. Xu H, Li B (2014) Dynamic cloud pricing for revenue maximization [J]. IEEE Trans Cloud Comput 1(2):158–171
13. Shi Y, Cheng J, Zhang J et al (2016) Smoothed Lp-minimization for Green cloud-RAN With User Admission Control [J]. IEEE J Sel Areas Commun 34(4):1022–1036
14. Masdari M, Salehi F, Jalali M et al (2017) A Survey of PSO-based Scheduling Algorithms in Cloud Computing [J]. J Netw Syst Manag 25(1):122–158
15. Wen Y, Zhang W, Luo H (2012) Energy-optimal Mobile Application Execution: Taming Resource-poor Mobile Devices with Cloud Clones. In: Proceedings of IEEE INFOCOM, vol PP, pp 2716–2720
16. Botta A, Donato WD, Persico V et al (2014) Integration of Cloud computing and Internet of Things: A Survey [C]// International Conference on Future Internet of Things and Cloud. IEEE Computer Society, pp 23–30
17. DÍaz M, MartÍn C, Rubio B (2016) State-of-the-art, Challenges, and Open Issues in the Integration of Internet of Things and Cloud Computing [J]. J Netw Comput Appl 67(C):99–117
18. Patel P, Ali MI, Sheth A (2017) On Using the Intelligent Edge for IoT Analytics [J]. IEEE Intell Syst 32(5):64–69
19. Daneels G, Municio E, Spaey K et al (2017) Real-time Data Dissemination and Analytics Platform for Challenging IoT Environments [C]// Global Information Infrastructure and NETWORKING Symposium. IEEE
20. Raafat HM, Hossain MS, Essa E et al (2017) Fog Intelligence for Real-time IoT Sensor Data Analytics [J]. IEEE Access PP(99):24062–24069
21. Jalali F, Hinton K, Ayre R et al (2016) Fog computing may help to save energy in cloud computing [J]. IEEE J Sel Areas Commun 34(5):1728–1739
22. Jalali F, Vishwanath A, Hoog JD et al (2016) Interconnecting Fog Computing and Microgrids for Greening IoT [C]// Innovative Smart Grid Technologies - Asia. IEEE, pp 693–698
23. Verma S, Kumar YA, Motwani D et al (2016) An Efficient Data Replication and Load Balancing Technique for Fog Computing Environment. In: Proceedings of 2016 3rd IEEE international conference on Computing for sustainable global development (INDIACom 2016), vol PP, pp 2888–2895
24. Xiao M, Wu J, Huang L et al (2015) Multi-task Assignment for Crowdsensing in Mobile Social Networks [C]// Computer Communications. IEEE, pp 2227–2235
25. Wen Z, Yang R, Garraghan P et al (2017) Fog orchestration for internet of things Services[J]. IEEE Internet Comput 21(2):16–24
26. Pham X-Q, Huh E-N (2016) Towards Task Scheduling in a Cloud-Fog Computing System. In: Proceedings of 18th Asia-Pacific Network Operations and Management Symposium (APNOMS 2016), PP, pp 1-4
27. Chen MH, Liang B, Dong M (2017) Joint Offloading and Resource Allocation for Computation and Communication in Mobile Cloud with Computing Access Point [C]// INFOCOM 2017 - IEEE Conference on Computer Communications, IEEE. IEEE, 159–66
28. Wang S, Huang X, Liu Y et al (2016) CachinMobile: An Energy-Efficient Users Caching Scheme for Fog Computing. In: Proceedings of 2016 IEEE/CIC International Conference on Communications in China (ICCC 2016), vol, PP, pp 1–6
29. Ni J, Lin X, Shen X (2018) Efficient and Secure Service-oriented Authentication Supporting Network Slicing for 5G-enabled IoT [J]. IEEE J Sel Areas Commun PP(99):1–14
30. Zhang JY, Wang ZJ, Quan Z et al (2018) Optimizing power consumption of mobile devices for video streaming over 4G LTE networks [J]. Peer-to-Peer Netw Appl 11(5):1101–1114
31. Boyd S, Vandenbergh L (2004) Convex optimization. Cambridge University Press, Cambridge
32. Valerio VD, Cardellini V, Presti FL (2013) Optimal pricing service provisioning strategies in cloud systems optimal a stackelberg game approach [C]// IEEE, International Conference on Cloud Computing. IEEE, pp 115–122
33. Soyata T, Muraleedharan R, Funai C et al (2012) Cloud-Vision: Real-time Face Recognition Using a Mobile-Cloudlet-Cloud Acceleration Architecture [C]// Computers and Communications. IEEE, pp 000059–000066
34. Yang L, Cao J, Cheng H et al (2015) Multi-User Computation partitioning for latency sensitive mobile cloud applications [J]. IEEE Trans Comput 64(8):2253–2266

35. Zakariayi S, Babaie S (2019) DEHCIC: a distributed energy-aware hexagon based clustering algorithm to improve coverage in wireless sensor networks [J]. Peer-to-Peer Netw Appl 12(4):689–704
36. Zhang L, Zhao GD, Zhou WL et al (2017) Primary channel gain estimation for spectrum sharing in cognitive radio networks[J]. IEEE Trans Commun 65(10):4152–4162

Huaiying Sun received her B.S. degree in computer science from Jiangxi University of Science and Technology in 2015. She is currently a PhD. Student in Computer Application Technology at East China University of Science and Technology (ECUST). Her research interests include software engineering, high confidence computing systems, cloud computing, service oriented computing, fog/edge computing and formal methods.

Huiqun Yu received his B.S. degree from Nanjing University in 1989, M.S. degree from East China University of Science and Technology (ECUST) in 1992, and PhD. degree form Shanghai Jiaotong University in 1995, all in computer science. He is currently a Professor of computer science with the Department of Computer Science and Engineering at ECUST. From 2001 to 2004, he was a Visiting Researcher in the School of Computer Science at Florida International University. His research interests include software engineering, high confidence computing systems, cloud computing and formal methods. He is a senior member of IEEE, China Computer Federation.

Guisheng Fan received his B.S. degree from Anhui University of Technology in 2003, M.S. degree from East China University of Science and Technology (ECUST) in 2006, and Ph.D. degree from East China University of Science and Technology in 2009, all in computer science. He is presently an Associate Professor of the Department of Computer Science and Engineering, East China University of Science and Technology. His research interests include formal methods for complex software system, service oriented computing, and techniques for analysis of software architecture.

Liqiong Chen received her B.S. degree from Anhui University of Technology in 2004, and Ph.D. degree from East China University of Science and Technology (ECUST) in 2009, all in computer science. She is presently an Associate Professor of the Department of Computer Science and Information Engineering, Shanghai Institute of Technology. Her research interests include formal methods for complex software system, service oriented computing, and techniques for analysis of software architecture.