IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON CONVERGENCE OF SENSOR NETWORKS, CLOUD COMPUTING, AND BIG DATA IN INDUSTRIAL INTERNET OF THING

Received December 29, 2017, accepted January 25, 2018, date of publication January 30, 2018, date of current version March 9, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2799548

A Mobile Cloud Based Scheduling Strategy for Industrial Internet of Things

CHAOGANG TANG[1], XIANGLIN WEI[2], SHUO XIAO[1], WEI CHEN[1], WEIDONG FANG[3], WUXIONG ZHANG[3], AND MINGYANG HAO[1]

[1]School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221000, China
[2]Nanjing Telecommunication Technology Research Institute, Nanjing 210000, China
[3]Key Laboratory of Wireless Sensor Network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai 200050, China

Corresponding author: Wei Chen (chenw@cumt.edu.cn)

This work was supported in part by the Jiangsu Province Natural Science Foundation of China under Grant BK20150201 and Grant BK20150193, in part by the National Natural Science Foundation and Shanxi Provincial People's Government Jointly Funded Project of China for Coal Base and Low Carbon under Grant U1510115, and in part by the Qing Lan Project and China Postdoctoral Science Foundation under Grant 2013T60574 and Grant 2016M601910.

ABSTRACT The Industrial Internet of Things is cited as the latest means for making manufacturing more flexible, cost effective, and responsive to changes in customer demands. In this paper, we present a mobile cloud based scheduling strategy for the industrial Internet of Things. Several computing paradigms, such as mobile cloud computing, fog computing, and edge computing can be integrated to the industrial Internet of Things, which allow to offload tasks to the cloud for execution. We model the task scheduling problem as an energy consumption optimization problem, while taking into account task dependency, data transmission, and some constraint conditions, such as response time deadline and cost, and further solve it by genetic algorithms. A series of simulation experiments are conducted to evaluate the performance of the algorithm and the results have shown that our proposal is more efficient than the baseline approach.

INDEX TERMS Industrial Internet of Things (IIoT), energy-efficient, mobile cloud computing, task scheduling.

I. INTRODUCTION

Mobile cloud computing (MCC), which combines wireless network and cloud computing and aims at improving the performance of mobile applications hosted at mobile devices such as PDAs and smartphones, has developed very fast in the past few years. Due to some inherent defects of mobile devices, e.g. low CPU speed, limited battery energy, insufficient storage space, and inadequate sensing capacities [3], mobile applications are confronted with many challenges in mobility management, quality of service (QoS) insurance, energy management and security issues. As a solution to these shortcomings, MCC succeeds in offloading some computing modules to be executed on powerful nodes in the cloud (e.g. cloudlet [19], [23]), which brings a few benefits against the traditional mobile services [12]. For example, with regards to some energy or resource-intensive mobile applications hosted in the mobile devices, offloading some parts of them to the remote cloud saves energy consumption greatly for the devices. Many mobile applications such as e-commerce, health-care, and computer games are developed under mobile cloud computing concept.

However, task offloading is not always efficient, since it depends on several factors, such as transmission bandwidth of the wireless channel, the energy consumption on task offloading at mobile devices, energy consumption on task execution at the cloud and so on. For example, mobility as the inherent attribute of the mobile devices may force mobile users to change the access point (AP) frequently when users move from one place to another. This kind of dynamics sometimes makes the wireless connection unavailable, thus rendering the waiting time longer than expected, which may degrade users Quality of Experience (QoE), even leading to users refusal to accept the response time especially for the urgent tasks. Besides, energy consumption is another important factor, which imposes great influence on offloading decision. For example, if the energy consumption caused by task offloading at mobile device and data transmission via wireless channel were larger than task execution locally without offloading, it would make no sense for tasks execution at cloud side remotely, from the viewpoint of saving power consumption for mobile devices. Most literatures about the task offloading and task scheduling in MCC model it as a multi-objective

(e.g. energy, cost, execution time) optimization problem, considering some constraints such as execution deadline. For example, for an urgent task, the total execution time should not go beyond users' specified deadline. However, most works assume that the tasks derived from an application are independent, which simplifies the uploading process, but does not always hold in MCC environments. For example, the tasks derived from partitioning the applications usually need some interaction such as data transmission among each other in order to perform their functions. In this paper, we model task uploading and scheduling as a multi-objective optimization problem with dependency relationships between tasks. We also take into account the data moving between tasks, as well as some constraints such as execution cost and execution time limitation imposed on performance metrics by mobile users. Specifically, each task within the application can be either uploaded to the mobile cloud or executed locally on the mobile device. Instinctively, tasks which need frequent interaction with mobile users are supposed to be performed at the mobile device, and tasks which need complicated computation and consume large energy on the other hand are supposed to upload to the mobile cloud.

The rest of the paper is organized as follows. In Section II, we review the related work on task scheduling in MCC. Section III addresses system model, formulates our optimization problem and section IV further proposes algorithms to solve this problem. Then, we conduct extensive experiments to verify the effectiveness and efficiency of our approach in Section V. Finally, the conclusion and future work are given in Section VI.

II. RELATED WORK

In this section, we view some current works about task scheduling problem in MCC. Usually, in order to reduce power consumption, speed up the execution of an application, or save storage space, the mobile application is partitioned into several pieces, knows as tasks, and then these tasks are partially scheduled onto the nodes for execution in the mobile cloud. The optimization objective mainly falls into two categories, either minimizing the total execution time also called, makespan, or minimizing the energy consumption [1], [2], [13], [17], [22], [38], [41]. Since the task scheduling problem is NP-hard [6], most works adopt heuristic approaches to solve this problem, which cannot guarantee to find the optimal solution, but it can find almost optimal solution.

Hung *et al.* [10], propose a task scheduling approach to guarantee a better accessibility to cloud network and speed up the processing time in MCC, taking into consideration some constraints such as the network bandwidth and cost for cloud usage. However, the details on how to obtain some metrics such as earliest start time or earliest finish time of tasks are not offered, and the algorithm complexity is unknown. Some works [4], [21], [24], [40] pay attention to the kinds of resources which the nodes in the MCC can provide, and schedule the tasks to the nodes in MCC combining it and the information on the amount and kinds of requested resources tasks need for execution, so as to find the most appropriate scheduling scheme. Wu *et al.* [22] proposed a task scheduling algorithm based on the quality of service (QoS) metrics, such as load balancing, average execution, and makespan. First, according to the QoS, they calculate the priorities of the tasks, and then tasks with higher priority are scheduled first on the nodes. Razaque *et al.* [18] proposed an efficient task scheduling algorithm for workflow allocation based on the availability of network bandwidth. For other methods, authors adopted Min-Min and Min-Max algorithms to assign tasks to each node in the cloud based on a nonlinear programing model. For the tasks obtained by partitioning the application, some are appropriate to be uploaded to the MCC while some are not. How to select suitable tasks to upload and guarantee that the task-precedence requirements and the application completion time constraint are satisfied has obtained a lot of attention in the past few years. Lin *et al.* [14] presented an algorithm, which started from a minimal-delay scheduling solution and then performs energy reduction by applying the dynamic voltage and frequency scaling technique.

Wang *et al.* [21], provided an energy-efficient dynamic offloading and resource scheduling policy to reduce energy consumption and shorten application execution time, so as to achieve energy-efficient computation offloading under the hard constraint for application completion time. Mahmood and Khan [16], proposed a greedy and a genetic algorithm (GA) with an adaptive selection of suitable crossover and mutation operations to allocate and schedule real-time tasks with precedence constraint on heterogamous virtual machines. There are also a number of works which deals with the task allocation and scheduling for real-time works in a cloud environments such as [5], [11], [15], [37], and [39]. Tsai *et al.* [20],proposed a differential evolution algorithm to schedule a set of tasks to minimize makspan and total cost, by embedding the Taguchi method within a differential evolution algorithm framework to exploit better solutions on the micro-space to be potential offspring.

In contrast to the aforementioned studies, we in this paper focus on minimizing the power consumption of mobile devices at the local side, while satisfying the task-precedence requirements and the application completion time constraint by genetic algorithms. For the optimization function, we describe it by mathematical model theoretically and then adopt the genetic algorithms to solve this problem.

Standing out from these heuristic approaches, GA has been applied to optimization problems in many fields, such as machine learning, pattern recognition, job scheduling and so on. Binary coding GAs (bAGs) dominate the early period of GA research, in which the simple representation, implementation and the outstanding theoretical literature [27] form the main incentives for the utilization of binary string chromosomes to represent the solution space. Additionally, more and more researches add substantial robustness and efficiency to their performance, such as gene expression GA [28], Linkage Learning

GA [29] and the Bayesian Optimization Algorithm [30]. On the other hand, the increasing application of GAs to continuous domain problems encourages the emergency and development of real coded GAs (rGAs), which use vectors of floating point numbers to represent and process the design and search space [31]. Intuitively, utilizing the floating point numbers to represent chromosomes is natural and easily understandable for some continuous domain problems and examples of applying rGAs to solve real world problems can be found in [32]–[34].

Examples of applying GAs to task scheduling in cloud computing and web service fields can be found in [4], [16], and [24], [35], [36]. Deng *et al.* [4], focus on the problem of service composition with temporal and QoS constraints in mobile cloud computing and aim to form such a service composition that not only satisfies both the time constraints and QoS constraints in a mobile service composition, but also ensures the composition to be executed successfully to the greatest extent in the uncertain mobile environment. Xu *et al.* [35], have proposed a multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing system, of which the basic idea is to exploit the advantages of both evolutionary and heuristic based algorithms while avoiding their drawbacks. Lin and Chong [36] presented a genetic algorithm (GA) based resource constraint project scheduling, incorporating a number of new ideas (enhancements and local search) for solving computing resources allocation problems in a cloud manufacturing system.

In order to improve resource utilization and task execution efficiency, a scheduling algorithm based on resource attribute selection (RAS) by sending a set of test tasks to an execution node to determine its resource attributes before a task is scheduled is proposed in [24], which selects the optimal node to execute a task according to its resource requirements and the fitness between the resource node and the task.

However, the approach proposed in this paper mainly has two points which differs from the aforementioned works. First, with regards to specific problems in different areas, objective function usually has different forms and concrete formulations. For example, we in this paper aim to minimize the power consumption at the mobile device while satisfy a series of constraints such as the tasks dependency relationships, the deadline of the expected execution time and the cost for executing the tasks in the mobile cloud. Second, the optimization works in this paper involve both binary and real variables, which gives rise to different chromosome representations, solution space and searching methods with concrete crossover and mutation operators. In the next section, we will detail our system model.

III. SYSTEM MODEL

A. SYSTEM DESCRIPTION

We consider a mobile cloud computing environment which mainly includes two parts, i.e., the mobile users and the mobile cloud, respectively. The mobile cloud architecture is shown in Fig. 1, in which tasks from mobile users are uploaded to the cloud via the wireless access point (AP). Here, an AP provides radio resources (e.g. bandwidth) and communication support. The incoming tasks are arranged for execution by a module called broker as shown in Fig. 1, which in the mobile cloud also performs an admission control by checking the availability of the computing resources such as CPU, memory and storage of the computing nodes, i.e., virtual machines (VMs).

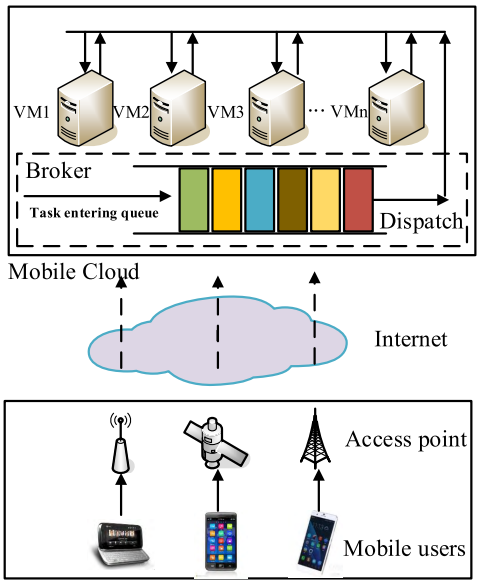


FIGURE 1. The mobile cloud architecture.

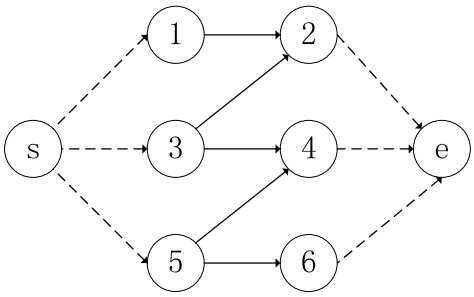


FIGURE 2. An example of application with task precedence relationships.

Usually, a mobile application consist of a set of tasks in different granularities, we denote the application by a directed acyclic task graph G = (V, E), as shown in Fig.2. Each node in G represents a task and an edge e(i, j) indicates the precedence relationship between tasks i and j, which means task j cannot start until the precedent task i completes. We add a virtual entry node (resp. exiting node) with dashed lines directed to the actual tasks (resp. by the actual tasks), so that in the task graph there is only one starting node and ending node, respectively. Note that, how to partition the application into tasks is not our focus, we pay our attention to the tasks uploading and task scheduling in this paper.

TABLE 1. Notation description.

Notation	Definition
γ_l	The processing speed of cores at the mobile device(MD)
γ_c	The processing speed of nodes at the clouds
T_t^l	The execution time when task t is executed locally
T_t^c	The execution time when task t is executed remotely
P_{idle}	Idle power of MDs
P_{active}	Running power of MDs
P_{trans}	Transmitting power of MDs
$S(t)$	The computation data size of task t
$\Delta m(t,s)$	The amount of data migration between task t and task s
E_t^l	The energy consumption of MDs when task t executing locally
E_t^c	The energy consumption of MDs when task t executing remotely
B	The bandwidth of the wireless channel
ρ	The price of computation unit provided by mobile clouds
T_{app}	The total execution time of the application

A task t can be defined as a 3-tuple $t = \langle tid, workload, \Delta m \rangle$ where *tid* denotes its identification in the application by numerical values, *workload* represents its computation amount when it is executed, and Δm denotes the amount of data migration between two tasks with precedence relationships.

If some tasks within an application are involved with uploading for execution, there are mainly two kinds of energy consumption at the mobile devices. For example, the mobile device needs to spend energy in executing local tasks of the application, denoted by computation consumption. Besides, the mobile device also needs to spend energy to transmit the remaining tasks to the MCC via the wireless channel such as WIFI, 3G/4G, and so on. We denote this kind of energy consumption by communication consumption. Obviously, if the sum of the two kinds of consumption is larger than the energy consumption spent by the mobile device when the entire application is executed locally, it makes no sense to have tasks uploaded to the MCC. Note that the energy consumption for mobile devices does not include the energy consumption by the mobile cloud, for the reason that we assume for each task uploaded to the mobile cloud for execution, the MCC will charge the mobile users, which is also the main distinction between cloud computing and grid computing. However, if the costs go beyond the mobile users budget, the mobile users may choose not to upload their tasks to the mobile cloud. For some urgent application, users usually specify a deadline, before which the result should be returned. Next, we will detail and formulate two kinds of energy consumptions and optimization function. To facilitate our further discussion, Table 1 lists the key notations of different types of energy consumptions in the proposed consumption model through the paper.

B. PROBLEM FORMULATION

Suppose the application consists of n tasks. For each task, there are two options to execute it, i.e., locally or remotely. Let φ_t be an execution indicator variable. $\varphi_t = 1$ if task t is executed at the mobile device and 0 otherwise. If it is executed locally, the execution time T_t^l can be calculated as follows:

$$T_t^l = \frac{S(t)}{\gamma_l} \tag{1}$$

The corresponding energy consumption of the mobile device is

$$E_t^l = \frac{S(t)}{\gamma_l} \cdot P_{active} \tag{2}$$

If t is executed remotely, the execution time can be calculated as follows:

$$T_t^c = \frac{S(t)}{\gamma_c} \tag{3}$$

The corresponding energy consumption of the mobile device is

$$E_t^c = \frac{S(t)}{\gamma_c} \cdot P_{idle} + \frac{S(t)}{B} \cdot P_{trans} \tag{4}$$

When the task is being executed in the cloud, the mobile device is in the idle state and we utilize P_{idle} to calculate the energy consumption of the mobile device. Accordingly, the total energy consumption of mobile devices accordingly can be written as follows:

$$E = \sum_{t \in V} (\varphi_t \cdot E_t^l + (1 - \varphi_t) \cdot E_t^c) + \frac{P_{trans}}{B} \cdot \sum_{(u,v) \in V} |\varphi_u - \varphi_v| \cdot \Delta m(u,v) \tag{5}$$

Considering the dependence relationships between tasks, we assume before a tasks t is about to be scheduled, all its immediate predecessor tasks should have already finished the execution, for the reason that t requires their output results as input parameters. In order to calculate the total execution time of the application, we introduce some definitions.

Definition 1 (Ready Time): The ready time of task t is defined as the earliest start time when all its immediate predecessor tasks have finished the execution. Thus, the ready time of task t which is executed locally on the mobile device, denoted by RT_t^l, is defined by

$$RT_t^l = \max_{s \in pred(m)} \max\{FT_s^l, FT_s^r\} \tag{6}$$

where $pred(m)$ denotes the set of the immediate predecessor tasks of task t. Note that for simplicity, we ignore the time consumption of input and output parameters transmission between mobile device and the cloud like the existing works [8], [14], due to the fact that the size of these parameters is much smaller than the task itself and the great power of the 4G network and WIFI also makes the time of data transmission neglected. If s is executed locally, $\max\{FT_s^l, FT_s^r\} = FT_s^l$ and $\max\{FT_s^l, FT_s^r\} = FT_s^c$ if s is scheduled to the cloud for execution. Thus, RT_t^l can be further rewritten as follows:

$$RT_t^l = \max_{s \in pred(m)} \{(1-\varphi_s)FT_s^l + \varphi_s FT_s^c\} \tag{7}$$

Similarly, the ready time of task t which is executed remotely on the cloud is defined by

$$RT_t^c = \max\{FT_t^{trans}, \max_{s \in pred(m)} FT_s^c\} \tag{8}$$

Definition 2 (Finish Time): The finish time of task t is defined as the time of task t which completely finishes execution. Thus, the finish time of task t which is executed locally on the mobile device, denoted by FT_t^l, is defined by

$$FT_t^l = RT_t^l + T_t^l \tag{9}$$

Similarly, the finish time of task t which is executed remotely on the cloud is defined by

$$FT_t^c = RT_t^c + T_t^c \tag{10}$$

Besides, we use FT_t^{trans} to represent the time when task t is completely uploaded to the cloud through the 4G network or WIFI. FT_t^{trans} can be defined as follows:

$$FT_t^{trans} = \max_{s \in pred(m)} \{FT_s^l, FT_s^{trans}\} + \frac{S(t)}{B} \tag{11}$$

By these definitions, we can calculate the execution time of the entire application as follows

$$T_{app} = \max_{t \in ending(app)} \{FT_t^c, FT_t^l\} \tag{12}$$

Where $ending(app)$ denotes the ending nodes of the application.

We assume that when two tasks with precedence relationships are scheduled to the same place, the communication energy consumption can be neglected. Intuitively, in order to reduce the energy consumption of the mobile device, the naive way is to offload the entire application to the cloud. However, it is generally not advisable for mobile users to follow this way for two reasons. Firstly, there are some applications which need frequent interactions with mobile users such as human face recognition, which could render serious communication energy consumptions and sometimes even degrade the mobile users quality of experience. Secondly, the difference between cloud computing and other computing patterns such as grid computing lies in that the cloud computing can earn its own profits. Therefore, mobile users are actually to enjoy the computing convenience by buying the services provided by the mobile cloud. We assume that the utilizing of mobile cloud resources is not free in this paper. Usually, the expenditure increases as the number of tasks uploaded to the cloud rises. Mobile users will not buy it if the costs go beyond their expectation. So the energy minimization problem can be formulated as follows:

$$(P) \quad f = Minimize\ E$$
$$S.t. \sum_{t \in V} \varphi_t \cdot \rho \cdot Size(t) \le Cost \tag{13}$$
$$T_{app} \le T_{deadline} \tag{14}$$
$$\varphi_t, \varphi_u \in \{0, 1\} \tag{15}$$

Where, $T_{deadline}$ represents the maximal latency that mobile users can tolerate. Inequation 13 means that the total expenditures should not go beyond users budget and Inequation 14 represents users time constraint. The variables and constants are listed in Table 1. Obviously, this scheduling problem is NP-complete. Moreover, P is a special linear optimization problem, since its objective function and constraint conditions contain absolute value symbols. An instinctive idea is to get rid of the absolute value in solving this optimization problem, which will be detailed in the next.

In order to solve the optimization problem P, we first transform this kind of problem to normal linear programming problem. Therefore, we proposed a two-step approach to solve problem P, i.e., normal linear programming transformation and constraint condition relaxation.

We introduce two other non-negative vectors $\theta = (\theta_1, \theta_2, \cdots, \theta_{|E|})$ and $\mu = (\mu_1, \mu_2, \cdots, \mu_{|E|})$. Consider the following optimization problem.

$$(Q) \quad Minimize\ f$$
$$= \sum_{t=1}^{t=N} (\varphi_t \cdot E_t^l + (1-\varphi_t) \cdot E_t^c) + \sum_{k=1}^{k=|E|} (\theta_k + \mu_k)$$
$$S.t. \sum_{t \in V} \varphi_t \cdot \rho \cdot Size(t) \le Cost \tag{16}$$
$$T_{app} \le T_{deadline} \tag{17}$$
$$\frac{P_{trans}}{B} \cdot (\varphi_u - \varphi_v) \cdot \Delta m(u, v) + \theta_k - \mu_k = 0$$
$$k = 1, 2, \cdots, |E|, \quad (u, v) \in E \tag{18}$$
$$\theta_k \ge 0, \quad \mu_k \ge 0 \tag{19}$$
$$\varphi_t, \varphi_u \in \{0, 1\} \tag{20}$$

For this optimization problem, the number of edges is $|E|$ and the number of tasks within the application is N which is equivalent to the problem statement in optimization problem P. For the constraint condition, we suppose that each edge corresponds to a unique edge identification, denoted by the edge number $k (k = 1, 2, \cdots, |E|)$, which is known in advance.

Theorem 1: If $(\varphi^*, \theta^*, \mu^*)$ is the best solution to problem Q, then we have $\theta^*(\mu^*)^T = 0$, and the value of the best

solution to optimization function is:

$$f^* = \sum_{t=1}^{t=N}(\varphi_t^* \cdot E_t^l + (1-\varphi_t^*) \cdot E_t^c) + \frac{P_{trans}}{B} \cdot \sum_{(u,v)\in V} \left|\varphi_u^* - \varphi_v^*\right| \cdot \Delta m(u,v)$$

Proof 1: Suppose that $\theta^*(\mu^*)^T \neq 0$, then there must exist at least one dimension j, such that $\theta_j^* > 0, \mu_j^* > 0$. When $\theta_j^* \leq \mu_j^*$, we can construct another solution as follows:

$$\varphi = \varphi^*$$

$$\hat{\theta}: \begin{cases} \hat{\theta}_j = 0 \\ \hat{\theta}_k = \theta_k^* \quad (k \neq j) \end{cases} \qquad \hat{\mu}: \begin{cases} \hat{\mu}_j = \mu_j^* - \theta_j^* \\ \hat{\mu}_k = \mu_k^* \quad (k \neq j) \end{cases}$$

Obviously, the new resulting answer $(\varphi^*, \hat{\theta}, \hat{\mu})$ is different from the best solution in the dimension j. For constraint condition 18, when k is equal to j,

$$\begin{aligned} &\frac{P_{trans}}{B} \cdot (\varphi_u^* - \varphi_v^*) \cdot \Delta m(u,v) + \hat{\theta}_j - \hat{\mu}_j \\ &= \frac{P_{trans}}{B} \cdot (\varphi_u^* - \varphi_v^*) \cdot \Delta m(u,v) + \mu_j^* - \theta_j^* \\ &= 0 \end{aligned}$$

Therefore, $(\varphi^*, \hat{\theta}, \hat{\mu})$ is also the solution to the problem Q. Besides, $\hat{f} - f^* = \hat{\theta}_j + \hat{\mu}_j - (\theta_j^* + \mu_j^*) = -2\theta_j^* < 0$, which contradicts the assumption that $(\varphi^*, \theta^*, \mu^*)$ is the best solution to problem Q. Therefore $\theta^*(\mu^*)^T = 0$ holds.

The way to prove that $\theta^*(\mu^*)^T = 0$ holds when $\theta_j^* > \mu_j^*$ is similar, so the proving is omitted here. Because $\theta^*(\mu^*)^T = 0$, so for each k $(k = 1, 2, \cdots, |E|)$, we have $\theta_k^* \mu_k^* = 0$. If $\theta_k^* = 0$, we have

$$\frac{P_{trans}}{B} \cdot (\varphi_u - \varphi_v) \cdot \Delta m(u,v) = \mu_k^* \geq 0$$

If $\mu_k^* = 0$, we have

$$\frac{P_{trans}}{B} \cdot (\varphi_u - \varphi_v) \cdot \Delta m(u,v) = -\theta_k^* \leq 0$$

Therefore, for both cases, we have

$$\begin{aligned} \theta_k^* + \mu_k^* &= \left|\frac{P_{trans}}{B} \cdot (\varphi_u^* - \varphi_v^*) \cdot \Delta m(u,v)\right| \\ &= \frac{P_{trans}}{B} \cdot \left|\varphi_u^* - \varphi_v^*\right| \cdot \Delta m(u,v) \end{aligned}$$

Therefore, **Theorem** 1 is proved.

Theorem 2: If problem P has a feasible solution, the problem Q also has a feasible solution and vice versa. Moreover, if Q has a best solution $(\varphi^*, \theta^*, \mu^*)$, then φ^* must be the best solution to P.

Proof 2: Suppose that φ is a feasible solution to problem P. We can define the vectors θ and μ as follows. When $\varphi_u - \varphi_v \geq 0$, let

$$\begin{cases} \theta_k = 0 \\ \mu_k = \dfrac{P_{trans}}{B} \cdot (\varphi_u - \varphi_v) \cdot \Delta m(u,v) \end{cases}$$

Otherwise, let

$$\begin{cases} \theta_k = -\dfrac{P_{trans}}{B} \cdot (\varphi_u - \varphi_v) \cdot \Delta m(u,v) \\ \mu_k = 0 \end{cases}$$

Where, $k = 1, 2, \cdots, |E|$. We can verify that (φ, θ, μ) is a feasible solution to problem Q, because it satisfy all the constraints. On the other hand, if (φ, θ, μ) is a feasible solution to problem Q, it is obvious that the vector φ is the solution to problem P. Assume that $(\varphi^*, \theta^*, \mu^*)$ is the best solution to problem Q, but φ^* is not the best solution to problem P. Then for problem P, there must exist the best solution, denoted by $\hat{\varphi}$, of which the value of the objective function $\hat{z}$ is smaller than that of φ^*. Namely,

$$\begin{aligned} \hat{z} &= \sum_{t\in V}(\hat{\varphi}_t \cdot E_t^l + (1-\hat{\varphi}_t) \cdot E_t^c) \\ &\quad + \frac{P_{trans}}{B} \cdot \sum_{(u,v)\in V} \left|\hat{\varphi}_u - \hat{\varphi}_v)\right| \cdot \Delta m(u,v) \\ &< z^* = \sum_{t\in V}(\varphi_t^* \cdot E_t^l + (1-\varphi_t^*) \cdot E_t^c) \\ &\quad + \frac{P_{trans}}{B} \cdot \sum_{(u,v)\in V} \left|\varphi_u^* - \varphi_v^*\right| \cdot \Delta m(u,v) \end{aligned}$$

Based on the vector $\hat{\varphi}$, we construct a feasible solution $(\hat{\varphi}, \hat{\theta}, \hat{\mu})$ to problem Q following the way above. Since $\hat{\theta}_k + \hat{\mu}_k = \frac{P_{trans}}{B} \cdot \left|\hat{\varphi}_u - \hat{\varphi}_v\right| \cdot \Delta m(u,v)$ holds, for $k = 1, 2, \cdots, |E|$, the value of the objective function with regards to $(\hat{\varphi}, \hat{\theta}, \hat{\mu})$ is

$$\begin{aligned} \hat{f} &= \sum_{t\in V}(\hat{\varphi}_t \cdot E_t^l + (1-\hat{\varphi}_t) \cdot E_t^c) + \sum_{k=1}^{|E|}(\hat{\theta}_k - \hat{\mu}_k) \\ &= \sum_{t\in V}(\hat{\varphi}_t \cdot E_t^l + (1-\hat{\varphi}_t) \cdot E_t^c) \\ &\quad + \frac{P_{trans}}{B} \cdot \sum_{(u,v)\in V} \left|\hat{\varphi}_u - \hat{\varphi}_v\right| \cdot \Delta m(u,v) \end{aligned}$$

According to **Theorem** 1, the optimization value of Problem Q is

$$f^* = \sum_{t\in V}(\varphi_t^* \cdot E_t^l + (1-\varphi_t^*) \cdot E_t^c) + \frac{P_{trans}}{B} \cdot \sum_{(u,v)\in V} \left|\varphi_u^* - \varphi_v^*\right| \cdot \Delta m(u,v)$$

Therefore, we have $\hat{f} < f^*$, which contradicts that $(\varphi^*, \theta^*, \mu^*)$ is the best solution to problem Q. Therefore, **Theorem** 2 is proved.

Now, for the optimization problem Q, we can further transform it. Let $\varphi = (\varphi_1, \varphi_2, \cdots, \varphi_n)$ be the uploading decision vector, $I_n = (I, I, \cdots, I)$ and $I_{|E|} = (I, I, \cdots, I)$ be respectively the vectors, of which each element is a unit matrix. Then, $I_n - \varphi = (I - \varphi_1, I - \varphi_2, \cdots, I - \varphi_n)$.

Let $E^l = (E_1^l, E_2^l, \cdots, E_N^l)$ be the energy consumption vector at the mobile device when tasks are executed locally and be the energy consumption vector at the mobile device when tasks are executed remotely. The optimization function f can be rewritten as follows:

$$\begin{aligned} f &= \sum_{t=1}^{t=N}(\varphi_t \cdot E_t^l + (1-\varphi_t)\cdot E_t^c) + \sum_{k=1}^{k=|E|}(\theta_k + \mu_k) \\ &= \varphi \cdot (E^l)^T + (I_N - \varphi)\cdot (E^c)^T + (\theta+\mu)\cdot (I_{|E|})^T \\ &= (\varphi, I_N - \varphi, \theta+\mu)\cdot (E^l, E^c, I_{|E|})^T \\ &= (\varphi_1, \cdots, \varphi_n, I-\varphi_1, \cdots, I-\varphi_n, \theta_1+\mu_1, \cdots, \theta_{|E|} \\ &\quad +\mu_{|E|})\cdot (E_1^l, E_2^l, \cdots, E_N^l, E_1^c, E_2^c, \cdots, E_N^c, I, \cdots, I)^T \\ &= (\varphi_1, \cdots, \varphi_n, I-\varphi_1, \cdots, I-\varphi_n, \theta_1, \cdots, \theta_{|E|}, \mu_1, \cdots, \\ &\quad \mu_{|E|})\cdot (E_1^l, E_2^l, \cdots, E_N^l, E_1^c, E_2^c, \cdots, E_N^c, I, \cdots, I)^T \\ &= (E_1^l, E_2^l, \cdots, E_N^l, E_1^c, E_2^c, \cdots, E_N^c, I, \cdots, I)\cdot (\varphi_1, \cdots, \\ &\quad \varphi_n, I-\varphi_1, \cdots, I-\varphi_n, \theta_1, \cdots, \theta_{|E|}, \mu_1, \cdots, \mu_{|E|})^T \end{aligned}$$

Thus, the problem Q can be a linear programming problem. In the next, we propose two different algorithms to solve this problem.

IV. DESIGNS AND SCHEDULING ALGORITHMS

It is an NP-complete problem to find the optimal upload decisions for the task scheduling. However, heuristic intelligent algorithm is a mature approach to get an optimal solution. In this paper, we adopt two approaches to solve this problem, i.e., the greedy algorithm and the genetic algorithm (GA).

A. PROPOSED GREEDY ALGORITHM

Greedy algorithms are best known for their simple implementation and speed, although they sometimes may not search out the best solution. However, the suboptimal solution always is acceptable if finding out the best solution is NP-hard and the time is exponential. In this paper, we first present the greedy algorithm, followed by the genetic algorithm. The pseudo code of our proposed greedy algorithm based task scheduling algorithm (GrABTS) is shown in Algorithm 1.

The algorithm first orders all the tasks based on the decreasing order of their computation data size, which can be done in advance as the input parameters. Intuitively, minimizing the power consumption at the mobile device means uploading the tasks with large computation amount to the mobile cloud for execution. A list denoted by $L_{schedule}$ is used to store the tasks which are upload to the mobile cloud for execution. The algorithm initializes $L_{schedule}$ by storing in sequence the tasks in T until the sum of expenditure spent in executing the tasks in $L_{schedule}$ in the mobile cloud exceeds the budget (lines 1-7). Then the algorithm calculates the total execution time of the application and verifies whether it exceeds the deadline. If the execution time goes beyond the deadline, the algorithm decides the replacement strategy as followings. First, it gets the task t_i in $L_{schedule}$ from back to front starting at the last position of $L_{schedule}$ denoted by $Pst_{-last}(L_{schedule})$ and task t_j in T from front to back starting at the next position of t_i denoted by $Pst_{-next}(t_i, T)$, respectively. Second, it recalculates T_{app} until the time constraint is satisfied (lines 8-19). Note that GrABTS sometimes may not find the best task scheduling solution and it usually just generates a suboptimal solution.

Algorithm 1 Greedy Algorithm Based Task Scheduling Algorithm (GrABTS)

Input :
T = tasks set sorted in the decreasing order of their computation data size
Output:
The optimal uploading decision

```
1  for i = 1; i < N; i++ do
2      t_i = i^th task in T
3      if ρ · Size(t_1) + Cost_cur < Cost then
4          L_schedule = L_schedule + t_i
5          set the corresponding uploading decision vector
           for t_i to 1
6      end
7  end
8  Calculate the initial power consumption E at the mobile
   device according to equation 5
9  Calculate T_app according to equation 12 based on
   L_schedule and φ
10 if T_app > T_deadline then
11     for i = Pst_last(L_schedule); i − −; i ≥ 0 do
12         for j = Pst_next(t_i, T); j++; j ≤ N do
13             Recalculate T_app by t_j instead of t_i
14             if T_app < T_deadline then
15                 Replace t_i by t_j in L_schedule
16                 update φ and E
17             end
18         end
19     end
20 end
```

B. PROPOSED GENETIC ALGORITHM

GAs are stochastic search techniques which are inspired by the principles of evolution and heredity [9] and described formally by Goldberg. GAs are robust algorithms for solving NP-hard global optimization problems, including scheduling problems. GAs are population based algorithms that work iteratively to obtain better solutions over the huge search space. In this paper, we propose a genetic algorithm based task scheduling (GABTS) to solve the optimization problem, of which the outline to describe the process is given in Algorithm 2. Note that our optimization problem is different from the traditional optimization problems which are solved by genetic algorithms, for the reason that the optimization function involves both binary variables (φ) and real variables(θ and μ).

Algorithm 2 Genetic Algorithm Based Task Scheduling (GABTS)

Input :
Parameters for genetic algorithm;
Parameters for task scheduling problem

Output:
The optimal uploading decision

```
Generate an initial population of chromosomes;
repeat
    Dispatch the tasks according to the execution indicator vector φ and evaluate the fitness value of each chromosome in the population;
    Select a part of chromosomes with a selection probability based on the fitness values of the chromosomes, denoted by p1;
    repeat
        Select two parent chromosomes from the individuals from the remaining chromosomes in the population randomly;
        With a crossover probability, do crossover operations on the pair of the selected individuals to form a new offspring;
        With a mutation probability, mutate new offspring at selected position in the chromosome;
    until the size of the population reaches the default;
    Select a part of the resulting population according to fitness values of the chromosomes, and combined with p1 to form a new population;
until end condition fulfilled;
```

In the next, we will elaborate on the details of the implementation of the genetic algorithm for task scheduling problem.

C. ENCODING AND INITIALIZATION

Taking into consideration the mixed variables in the optimization objective, for the energy-efficient and constraints-satisfied task scheduling problem in MCC, we represent an individual (i.e. a chromosome) in the population of our algorithm by a $3 \times N$ matrix as shown in Table 2, in which the first row is the execution indicator vector $\varphi_i (\in \{0, 1\})$ to denote whether the task is uploaded or not, and the next two rows are the introduced non-negative auxiliary variables in Problem Q.

TABLE 2. Chromosome representing in GABTS.

φ_1	φ_2	$\cdots$	φ_n
θ_1	θ_2	$\cdots$	θ_n
μ_1	μ_2	$\cdots$	μ_n

The initial population consists of ps randomly generated individuals, where ps is the size of the population which is kept as a constant through the generations. According to different purposes, we can tune the value of ps in the preliminary computational experiments.

D. FITNESS FUNCTION

Fitness value as a metric to evaluate the individuals decides which individuals would be used to generate the next generate population, so the design of the fitness function is crucial to the genetic algorithm, which exercises a great influence on the speed of searching out the best solution to the optimization problem.

In this paper, we aim to minimize the energy consumption of the mobile devices by uploading some parts of tasks to the cloud. Hence the fitness function is the same as the optimization object function denoted by problem Q.

E. SELECTION OPERATOR

Selection is an important part of genetic algorithm since it has a significant impact on the convergence of the genetic algorithm. Intuitively, the better fitted an individual, the larger the probability of its survival. To this end, there are many strategies and rules, such as the roulette-wheel selection [7], which assumes that the probability that an individual is selected to produce the next generation is proportional to its fitness value. In each generation, we select a portion of individuals from the current generation according to the selection probability. We repeatedly select two individuals as parents from the remaining individuals at random and further generate new offspring by crossover and mutation operation. Then we combine the resulting offspring and the selected individuals based on selection probability to form the next generation.

F. CROSSOVER OPERATOR

The crossover operator is to generate new offspring by exchanging gene segment of two selected parents. Since the chromosome is represented by mixed variables, for the crossover operator we need to apply different methods to deal with the binary and real variables.

1) CROSSOVER FOR BINARY VARIABLE φ

In the task scheduling problem, we adopt single-point crossover to generate the offspring. Specifically, we select randomly a crossover position to divide the parent chromosome into two segments, with regards to the first row in the chromosome. Then for the two selected parents, we exchange the corresponding gene segment, e.g., smaller gene segment, to form the new offspring, as shown in Figure 3.

2) CROSSOVER FOR REAL VARIABLES θ AND μ

For crossing the real variables θ and μ of the proposed GA, we adopt the simulated binary crossover operator [26] to generate two children solutions from two parent solutions as follows:

$$\begin{cases} \theta_k^{c_1} = \frac{1}{2} \cdot [(1+\overline{\beta}) \cdot \theta_k^{p_1} + (1-\overline{\beta}) \cdot \theta_k^{p_2}] \\ \theta_k^{c_2} = \frac{1}{2} \cdot [(1-\overline{\beta}) \cdot \theta_k^{p_1} + (1+\overline{\beta}) \cdot \theta_k^{p_2}] \end{cases}$$

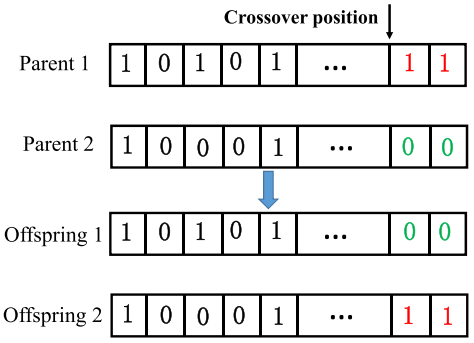


FIGURE 3. Crossover operation for binary variable φ.

where θ_k^{ci} and θ_k^{pi} $(i = 1, 2)$ denote the children and parents, respectively. From equation 18, we observe that although θ_k and μ_k are dependent on each other, θ_k still can have estimated maximum value $max(\theta_k)$. For example, $max(\theta_k) = max(\mu_k) + \Delta m(u, v) \cdot \frac{P_{trans}}{B}$, when $\varphi_u = 0, \varphi_v = 1$. We here assume the each element θ_k in vector θ follows the normal distribution by setting a upper bound for θ. For $\overline{\beta}$, we can be designed it as follows:

$$\overline{\beta} = \begin{cases} (2\xi)^{\frac{1}{1+n}} & \xi \leq 0.5 \\ (\frac{1}{2(1-\xi)})^{\frac{1}{1+n}} & otherwise; \end{cases}$$

where $\xi (0 < \xi < 1)$ is a random number uniformly distributed in [0, 1] and n is the number of tasks. We can observe from these crossover design that the search power of the crossover operator, i.e, a measure of how flexible the crossover operator is to create an arbitrary point in the search space, is sufficient to satisfy a number of criteria suggested by Radcliffe. The crossover operator for vector μ can be solved the same way as mentioned above.

G. MUTATION OPERATOR

In order to cover more extensive search space, we use a variable to control the mutation probability when the mutation position is chosen at random. Mutation operator usually alters a gene position locally to hopefully generate a better offspring.

1) MUTATION FOR BINARY VARIABLE φ

For the binary execution decision vector φ in the task scheduling problem, we turn 0 into 1 and vice versa at the chosen mutation position. Even though the bad offspring may be created, they will still be eliminated by repeated selection operation based on the fitness function in the next generation.

2) MUTATION FOR REAL VARIABLES θ AND μ

We apply the polynomial mutation operator [25] to the mutating operator for the real variables of the chromosome. Generally, polynomial mutation uses a polynomial probability distribution to make a continuous variable (e.g. θ and μ) changed from the current value to a neighboring value. The distribution generates the next generation offspring by a function of the distribution index η. Specially, the details on mutation operator are described as follows:

$$\delta_1 = \frac{\theta_k^p - min(\theta_k^p)}{max(\theta_k^p) - min(\theta_k^p)}$$

$$\delta_2 = \frac{max(\theta_k^p) - \theta_k^p}{max(\theta_k^p) - min(\theta_k^p)}$$

$$\delta = \begin{cases} [2r + (1-2r)(1-\delta_1)^{\eta+1}]^{\frac{1}{\eta+1}} - 1 & r \leq 0.5 \\ 1 - [2(1-r) + 2(r-0.5)(1-\delta_2)^{\eta+1}]^{\frac{1}{\eta+1}} & \\ & otherwise, \end{cases}$$

$$\theta_k^c = \theta_k^p + \delta(max(\theta_k^p) - min(\theta_k^p))$$

where, r is a random number uniformly distributed in [0, 1] and η is the index for polynomial mutation. θ_k^c and θ_k^p denote the children and parents, respectively. $max(\theta_k^p)$ (resp. $min(\theta_k^p)$) denote the upper (resp. lower) bound of the solution variable.

V. SIMULATION AND RESULTS ANALYSIS

We present in this section the experiments via numerical simulations to evaluate the effectiveness and efficiency of our approach.

TABLE 3. Parameter settings in GABTS.

Name	Value	Default Value
Population size	[100,1000]	500
Crossover probability	[0.3,0.8]	0.5
Mutation probability	[0.01,0.1]	0.05
Maximum iterations	2000	2000
Chromosome length	[10,30]	20

A. EXPERIMENTAL SETUP

We run our experiments on a laptop with 2.5GHz Intel CPU, 8192M of RAM, Microsoft Win7 Operating System. The algorithms are implemented in C++ and evaluated under different parameter settings, such as selection rate and mutation rate in GA. We initialize the task graph structure (i.e., DAG) based on both real-world application and random simulation. For each task in the constructed workflow, we generate randomly the corresponding workload and transmission data which are transmitted into the subsequent tasks. For GA related parameter settings, we list them in Table 3. For example, for the crossover probability, we vary it from 0.3 to 0.8 with a step of 0.05 and the mutation probability from 0.01 to 0.1. In our experiments, once the population is initialized, for each task (i.e., gene value in each chromosome), the decision to upload it or not is confirmed. In order to simplify the system model and avoid calculating waiting time among tasks in the process of uploading, we assume that each task to be uploaded to the mobile cloud does not need waiting. Sometimes it is necessary for massive calculation and achievable via offline technique.

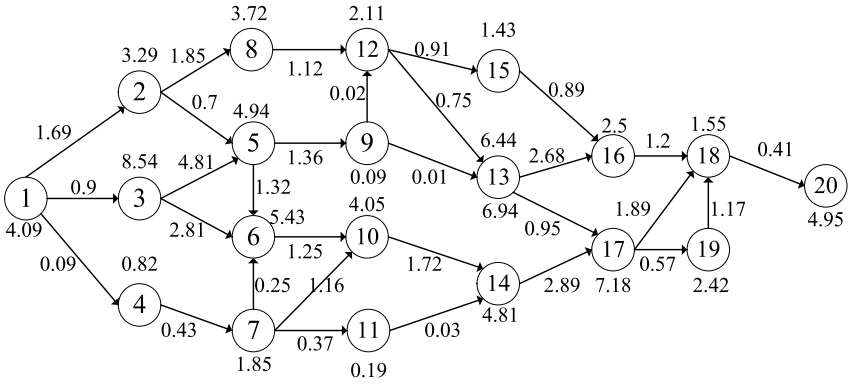


FIGURE 4. Example of an application with 20 tasks.

When we construct the task graph based on the real-world application, the task graph usually has certain structure, number of nodes and task dependency relationships. For example, we conduct the first set of experiments based on the task graph shown in Fig. 4 to validate the influence of crossover probability and mutation probability on the rate of convergence in GA. In Fig. 4, there are 20 tasks in the application, each numerical value around the node representing the computation data size of the task and each numerical value on the edge representing the amount of data migration between two tasks with dependency relationships. Note that the workload of each task and the amount of data transmission between two tasks with dependency relationships are generated randomly.

B. EXPERIMENTAL RESULTS

In this section, we report the experimental results in two ways. First, we evaluate the influence of parameter settings on the performance of GABTS. For example, different crossover probabilities and mutation probabilities may give rise to different iterations when achieving the best solution. On the other hand, we also need to validate the influence of the number of tasks in the application on the performance of GABTS with regards to average makespan and iterations when achieving the best solution. Second, we compare our approach with other approaches and further analyze the comparison results comprehensively.

1) PARAMETERS INFLUENCE ON GABTS

First, we study the influence of the mutation probability (MP) on the makespan of finding the optimal solution with GA. According to parameter settings, we first set the crossover probability (CP) to the default value and maximal iteration (MI) to 500. Note that we run the algorithm 50 times under each mutation probability to obtain the average makespan and the results are shown in Fig.5.

From Fig.5, we can see that the makespan increases when the mutation probability either increases or decreases from 0.02, which means that the total execution time of the application reaches the minimum when the algorithm finds the best solution, i.e., achieving the minimum energy consumption (MEC) with mutation probability being 0.02. A striking conclusion is that the average makespan almost increases by 67%, when the mutation probability is set to 0.02 and 0.1, respectively. Therefore, it is crucial to set the appropriate mutation probability to solve our task scheduling problem. The corresponding iterations versus mutation probability is shown in Fig.6. We can see that there are no obvious correlations between iterations and MEC when mutation probability varies. When mutation probability ranges from 0.02 to 0.1,

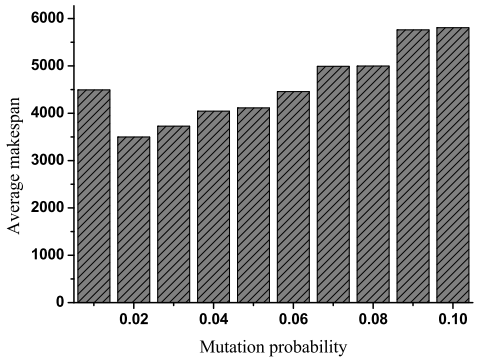


FIGURE 5. The average makespan under different MP.

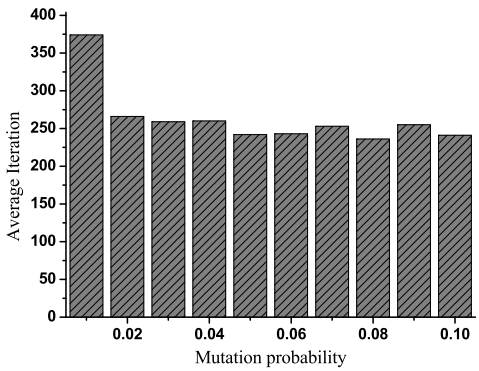


FIGURE 6. The average iterations when achieving MEC.

the average iterations to reach the minimum energy consumption is roughly the same. It is interesting that the number of average iterations when mutation probability is 0.01 is much bigger than others. The main reason is that a lower mutation probability gives rise to less coverage of population. Therefore, achieving the best solution, i.e., the minimum energy consumption, usually takes much more iterations than other situations.

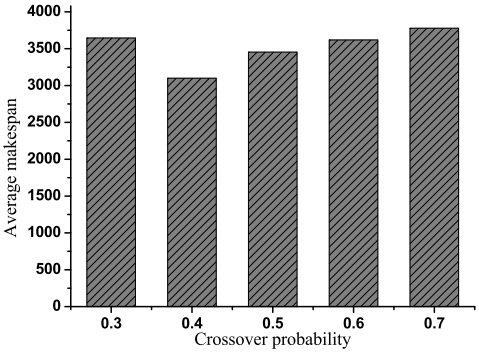


FIGURE 7. The average makespan under different CP.

We conduct the next set of experiments to validate the effects of crossover probability on the makespan when achieving the minimum energy consumption, which is shown in Fig.7. We set the mutation probability to 0.02 according to the experimental results in Fig.5. The maximum iteration is still 500. It is easily noted that the fastest way to achieve the minimum energy consumption is to set crossover probability to 0.4 compared to other crossover probability settings. The average makespan increases by 20% when crossover probability is set to 0.4 and 0.7, respectively. Therefore, the same as mutation probability, it is also crucial to choose appropriate crossover probability to solve the task scheduling problem in this paper.

In order to verify the influence of the number of tasks on finding out the best solution (i.e., the minimum energy consumption) with regards to average makespan and iterations, we conduct the following experiments. First, we generate the task graph randomly, and the number of tasks ranges from 10 to 19. For other parameter settings, the population size is set to 100, the maximum generations 1000, the mutation probability 0.05 and the crossover probability 0.5. The result is depicted in Fig.8, where the left Y-axis represents the average makespan while the right Y-axis represents the corresponding iterations when achieving the minimum energy consumption. It is clear that the number of tasks have significant influence on the time to find the best solution. For example, when the number of tasks increases from 10 to 19, the corresponding average makespan have almost increased by 230%. It is understandable, for the reason that the execution time of tasks either at the mobile device or at the cloud size increases as well as the communication time, not mention to the task dependency relationships that seriously restrict the concurrent execution of related tasks. On the other

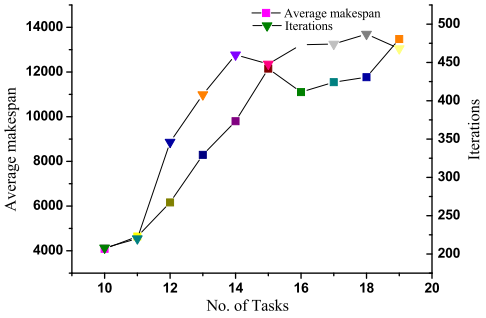


FIGURE 8. The average makespan under different No. of tasks.

hand, the number of iterations of GA also increases when the number of tasks increases. The increment of number of tasks means the increment of the length of chromosome, which lead to a more extensive genetic diversity of population. As a result, the number of iterations increase.

2) PERFORMANCE COMPARISON

In this section, we will present comparative performance of GABTS with other similar algorithms. We choose the simple GAs as the benchmark. Namely, we apply simple genetic algorithm to solving problem P. After introducing two vectors θ and μ to convert the optimization problem from problem P to problem Q, we use the hybrid genetic algorithm which is presented in IV-C and involved with both binary variable and real variable to solving the problem. We compare the approach with both the simple genetic algorithm and the proposed greedy approach respectively in this section.

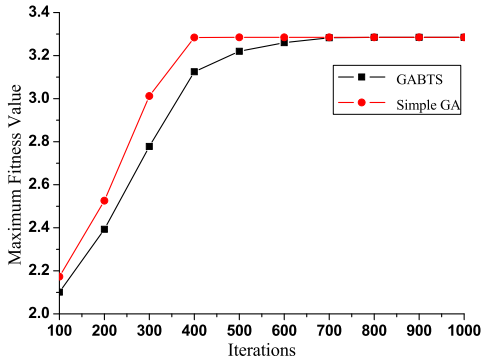


FIGURE 9. Performance comparison (Simple GA vs. GABTS).

Based on the aforementioned experimental results about the influence of GA parameters on the performance of GABTS, we conduct the following experiments, where we compare the GABTS to the simple genetic algorithms with regards to problem P and problem Q, respectively. The results are shown in Fig. 9, where we set CP to 0.4, MP to 0.02 and the length of chromosome to 20, respectively. Note that, for the upper and lower bound of each element in the introduced vectors θ and μ, we estimate them by equation 18. From Fig. 9 we can observe that GABTS achieves the maximum

fitness value faster than simple genetic algorithem. Specially, when the number of iterations reaches 400, GABTS can achieve the best fitness value, while the simple GA hangs around with the suboptimal fitness value until the number of iterations reaches 700. Therefore, when the number of iteration ranges from 100 to 700, GABTS has a great advantage against the simple GA. Although GABTS involves both binary and real variables, which makes the chromosome representation more complicated compared to simple binary coding chromosome, the efficiency of searching out the best solution is still acceptable.

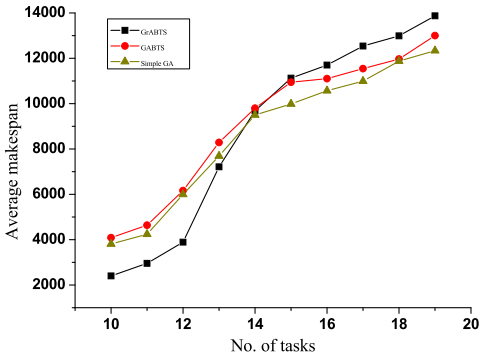


FIGURE 10. Performance comparison under different approaches.

The experimental results for evaluating the performance of simple GA, GrABTS and GABTS as shown in Fig.10, in which for genetic algorithms (simple GA and GABTS), the population size is set to 100, the maximum generations 1000, the mutation probability 0.05 and the crossover probability 0.5. We can observe that as the number of tasks involved in the application increases, the performance of GrABTS degrades sharply compared to GABTS and simple GA. When the number of tasks is small, GrABTS presents a great advantage against simple GA and GABTS, e.g. when the number of tasks is smaller than 14. As expected, the execution time of simple GA is relatively smaller than that of GABTS due to the complicated representation and processing of chromosome, crossover and mutation operators. However, we also notice that among the three approaches, GABTS are the most likely to find the optimal solution while simple GA and GrABTS sometimes can only find the suboptimal solution.

VI. CONCLUSION

Task scheduling is known as an NP-hard problem, which has attracted lots of attention in the past few years. The task scheduling under the context of mobile cloud computing has its own characteristics. For example, power consumption at the mobile device usually restricts the deployment and utilization of massive and complicated application at the mobile device. Hence, tasks are scheduled to the mobile cloud for execution is an efficient way to save power consumption of the mobile device.

We in this paper models this kind of task scheduling problem as an energy consumption optimization problem, while taking into account task dependency, data transmission and some constraint conditions such as response time deadline and cost, and further solve it by genetic algorithms. For the future work, we will test the performance of algorithms with much larger task graphs and devise more efficient heuristic algorithms to solve this task scheduling problem.

REFERENCES

[1] A. I. Awad, N. A. El-Hefnawy, and H. M. Abdel_Kader, "Enhanced particle swarm optimization for task scheduling in cloud computing environments," *Proc. Comput. Sci.*, vol. 65, pp. 920–929, Jan. 2015.

[2] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *J. Syst. Softw.*, vol. 99, pp. 20–35, Jan. 2015.

[3] M. Conti *et al.*, "Research challenges towards the Future Internet," *Comput. Commun.*, vol. 34, no. 18, pp. 2115–2134, 2011.

[4] S. Deng, L. Huang, H. Wu, and Z. Wu, "Constraints-driven service composition in mobile cloud computing," in *Proc. IEEE Int. Conf. Web Serv.*, Jun. 2016, pp. 228–235.

[5] S. Deniziak, L. Ciopinski, G. Pawinski, K. Wieczorek, and S. Bak, "Cost optimization of real-time cloud applications using developmental genetic programming," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput.*, Dec. 2014, pp. 774–779.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA, USA: Freeman, 1979.

[7] David E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. vol. 7. Boston, MA, USA: Addison-Wesley, 1989, pp. 2104–2116.

[8] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[9] J. H. Holland. *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.

[10] P. P. Hung, T.-A. Bui, and E.-N. Huh, "A new approach for task scheduling optimization in mobile cloud computing," in *Frontier and Innovation in Future Computing and Communications* (Lecture Notes in Electrical Engineering), vol. 301. Berlin, Germany: Springer, 2014, pp. 211–220.

[11] K. H. Kim, A. Beloglazov, and R. Buyya, *Power-Aware Provisioning of Virtual Machines for Real-Time Cloud Services*. Hoboken, NJ, USA: Wiley, 2011.

[12] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[13] Y. C. Lee, C. Wang, Y. Albert Zomaya, and B. B. Zhou, "Profit-driven scheduling for cloud services with data access awareness," *J. Parallel Distrib. Comput.*, vol. 72, no. 4, pp. 591–602, 2012.

[14] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 175–186, Mar./Apr. 2015.

[15] S. Liu, G. Quan, and S. Ren, "On-line preemptive scheduling of real-time services with profit and penalty," in *Proc. IEEE Southeastcon*, Mar. 2011, pp. 1476–1481.

[16] A. Mahmood and S. A. Khan, "Hard real-time task scheduling in cloud computing using an adaptive genetic algorithm," *Computers* vol. 6, no. 2, p. 15, 2017.

[17] S. K. Panda, I. Gupta, and K. Prasanta Jana, "Allocation-aware task scheduling for heterogeneous multi-cloud systems," in *Proc. Int. Symp. Big Data Cloud Comput. Challenges*, 2015, pp. 176–184.

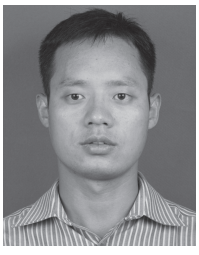
[18] A. Razaque, N. R. Vennapusa, N. Soni, G. S. Janapati, and K. R. Vangala, "Task scheduling in cloud computing," in *Proc. IEEE Long Island Syst., Appl. Technol. Conf.*, Apr. 2016, pp. 1–5.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

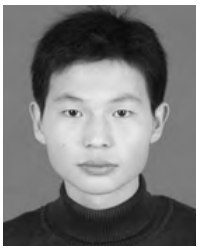
[20] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Comput. Oper. Res.*, vol. 40, pp. 3045–3055, Dec. 2013.

[21] J. Wang, J. Tang, G. Xue, and D. Yang, "Towards energy-efficient task scheduling on smartphones in mobile crowd sensing systems," *Comput. Netw.*, vol. 115, pp. 100–109, Mar. 2017.

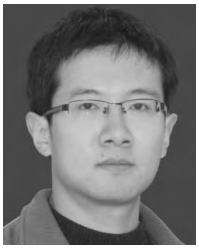
[22] X. Wu, M. Deng, R. Zhang, B. Zeng, and S. Zhou, "A task scheduling algorithm based on QoS-driven in cloud computing," *Proc. Comput. Sci.*, vol. 17, pp. 1162–1169, Jan. 2013.
[23] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *Proc. IEEE 38th Conf. Local Comput. Netw.*, Oct. 2014, pp. 589–596.
[24] Y. Zhao, L. Chen, Y. Li, P. Liu, X. Li, and C. Zhu, "RAS: A task scheduling algorithm based on resource attribute selection in a task scheduling framework," in *Proc. Int. Conf. Internet Distrib. Comput. Syst.*, 2013, pp. 106–119.
[25] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Inf.*, vol. 26, no. 4, pp. 30–45, 1996.
[26] D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. Hoboken, NJ, USA: Wiley, 2001.
[27] D. E. Goldberg, *The Design of Innovation: Lessons From and for Competent Genetic Algorithms*. Norwell, MA, USA: Kluwer, 2002.
[28] H. Kargupta, "The gene expression messy genetic algorithm," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 814–819.
[29] G. R. Harik and D. E. Goldberg, "Linkage learning through probabilistic expression," *Comput. Methods Appl. Mech. Eng.*, vol. 186, nos. 2–4, pp. 295–310, 2000.
[30] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz, "BOA: The Bayesian optimization algorithm," in *Proc. 1st Annu. Conf. Genetic Evol. Comput.*, 1999, pp. 525–532.
[31] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis," *Artif. Intell. Rev.*, vol. 12, no. 4, pp. 265–319, 1998.
[32] I. G. Damousis, A. G. Bakirtzis, and P. S. Dokopoulos, "Network-constrained economic dispatch using real-coded genetic algorithm," *IEEE Trans. Power Syst.*, vol. 18, no. 1, pp. 198–205, Feb. 2003.
[33] S. Panda and C. Ardil, "Real-coded genetic algorithm for robust power system stabilizer design," *Int. J. Electr., Comput. Syst. Eng.*, vol. 2, no. 1, pp. 6–14, 2008.
[34] M. S. Arumugam, M. V. C. Rao, and R. Palaniappan, "New hybrid genetic operators for real coded genetic algorithm to compute optimal control of a class of hybrid systems," *Appl. Soft Comput.*, vol. 6, no. 1, pp. 38–52, 2005.
[35] Y. Xu, K. Li, T. T. Khac, and M. Qiu, "A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems," in *Proc. IEEE Int. Conf. High Perform. Comput. Commun.*, Jun. 2012, pp. 639–646.
[36] Y.-K. Lin and C. S. Chong, "Fast GA-based project scheduling for computing resources allocation in a cloud manufacturing system," *J. Intell. Manuf.*, vol. 28, no. 5, pp. 1189–1201, 2015.
[37] C. Zhu, L. Shu, V. C. M. Leung, S. Guo, Y. Zhang, and L. T. Yang, "Secure multimedia big data in trust-assisted sensor-cloud for smart city," *IEEE Commun. Mag.*, vol. 55, no. 12, pp. 24–30, Dec. 2017.
[38] C. Zhu, H. Zhou, V. C. M. Leung, K. Wang, Y. Zhang, and L. T. Yang, "Toward big data in green city," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 14–18, Nov. 2017.
[39] C. Zhu, V. C. M. Leung, K. Wang, L. T. Yang, and Y. Zhang, "Multi-method data delivery for green sensor-cloud," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 176–182, May 2017.
[40] C. Zhu, X. Li, V. C. M. Leung, L. T. Yang, E. C.-H. Ngai, and L. Shu, "Towards pricing for sensor-cloud," *IEEE Trans. Cloud Comput.*, to be published.
[41] C. Zhu, J. J. P. C. Rodrigues, V. C. M. Leung, L. Shu, and L. T. Yang, "Trust-based communication for industrial Internet of Things," *IEEE Commun. Mag.*, to be published.



CHAOGANG TANG received the B.S. degree from the Nanjing University of Aeronautics and Astronautics in 2007 and the Ph.D. degree from the School of Information Science and Technology, University of Science and Technology of China in 2012. He is currently a Lecturer with the China University of Mining and Technology. His research interests include mobile cloud computing, fog computing, Internet of Things, big data, and WSN.



XIANGLIN WEI was born in Anhui, China. He received the bachelor's degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2007, and the Ph.D. degree from the University of Science and Technology, Nanjing, China, in 2012. He is currently a Researcher with the Nanjing Telecommunication Technology Research Institute, Nanjing, China. His research interests include cloud computing, peer-to-peer network, network anomaly detection, network measurement, and distributed system design and optimization. He has served as an editorial member of many international journals and a TPC member of a number of international conferences. He has also organized a few special issues for many reputed journals.

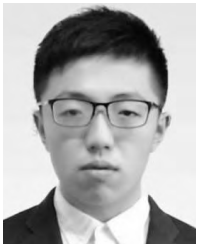


SHUO XIAO received the Ph.D. degree in traffic information engineering and control from Beijing Jiaotong University in 2010. Since 2010, he has been with the China University of Mining and Technology, where he is currently an Associate Professor. His research interests include wireless sensor networks, measure and control systems.

WEI CHEN received the Ph.D. degree in communications and information systems from the China University of Mining and Technology at Beijing, Beijing, China, in 2008. In 2008, he joined the School of Computer Science and Technology, China University of Mining and Technology at Xuzhou, where he is currently a Professor. His research interests include machine learning, image processing, and wireless communications.

WEIDONG FANG received the B.E. degree in industrial electrical automation from Shandong University, Jinan, China, in 1993, the M.E. degree in communication and electronic systems from the China University of Mining and Technology, Beijing, China, in 1998, and the Ph.D. degree in electromagnetic fields and microwave techniques from Shanghai University, Shanghai, China, in 2016. He is currently as an Associate Professor with the Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China. His current research interests include energy efficiency and information security in wireless sensor network, trust management, secure network coding, and secure routing protocol.

WUXIONG ZHANG received the B.E. degree in information security from Shanghai Jiao Tong University, Shanghai, China, in 2008, and the Ph.D. degree in communication and information systems from the Shanghai Institute of Microsystem and Information Technology (SIMIT), Chinese Academy of Sciences, Shanghai, in 2013. He is currently an Associate Professor with SIMIT and is serving as an Associate Professor with the Shanghai Research Center for Wireless Communications. His research interests include beyond third-generation mobile communication systems and vehicular networks.



MINGYANG HAO received the B.S. degree in electronic information science and technology from the Shandong University of Science and Technology in 2016. He is currently pursuing the Ph.D. degree with the China University of Mining and Technology, where he is involved in data analysis related learning and research.