IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received November 13, 2020, accepted November 22, 2020, date of publication November 26, 2020,
date of current version December 14, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3040785

EERA: An Energy-Efficient Resource Allocation Strategy for Mobile Cloud Workflows

JUAN LI[1,2] **AND XIAOLU XU**[3,4]
[1]School of Computer Science and Engineering, Wuhan Institute of Technology, Wuhan 430205, China
[2]Hubei Key Laboratory of Intelligent Robot, Wuhan Institute of Technology, Wuhan 430205, China
[3]Wuhan NARI Ltd. Liability Company of State Grid Electric Power Research Institute, Wuhan 430074, China
[4]State Grid Electric Power Research Institute, NARI Group Corporation, Nanjing 211000, China

Corresponding author: Juan Li (ljwuse2012@whu.edu.cn)

This work was supported in part by the Hubei Province Natural Science Foundation of China under Grant 2019CFB172, and in part by the Wuhan Institute of Technology, China, through the Science Foundation Research Project, under Grant K202035.

ABSTRACT Mobile cloud computing (MCC) which can invoke cloud services and offload tasks from the mobile device to the cloud has become an appropriate computing paradigm to provide many useful and complex workflow applications. However, offloading tasks needs extra communication time and energy, which leads to a conflict between saving the energy and improving the QoS. Thus, how to allocate these different resources (mobile or cloud) efficiently during the workflow execution process to optimize the system energy consumption under the deadline constraint is still a huge challenge in MCC. To address such an issue, this article proposes an energy-efficient resource allocation strategy for workflow applications in MCC. Firstly, we formulate the resource allocation problem into an optimization problem with the goal of minimizing the UtilityCost which represents the trade-off between energy consumption and the total execution time. Then, we use an energy-efficient resource allocation algorithm shortened as EERA to solve the problem. The algorithm firstly finds all the Partial critical paths of the workflow graph to segment the whole optimization problem into several local problems, and then uses the discrete particle swarm optimization algorithm to find the local optimal solution for every local problem, finally the local optimal solution is integrated to obtain the global one. In addition, we have adapted several representative algorithms for evaluation and comparison purpose. Comprehensive experimental results show that our EERA based strategy achieves the best overall performance on three key measurements including the energy consumption of the device, the execution time of the workflow and the UtilityCost.

INDEX TERMS Workflow, resource allocation, energy saving, task offloading, mobile cloud computing.

I. INTRODUCTION

Mobile devices, such as smartphones and tablets, are rapidly becoming the preferred choice of both organizational and personal computing due to their super convenience in time and location. Cisco VNI report (2017-2022) [1] shows that the average amount of traffic per smartphone in 2017 grew 49 percent and had reach to 2.3 GB per month. Along with the 5G development, it has also predicted that by 2022, a 5G connection will generate 2.6 times more traffic than the average 4G connection. In the meantime, a large number of innovative mobile applications which need increasingly large computing power are being produced in the market [2]. However, issues such as limited computing power and storage, low bandwidth and limited battery life are still hindering the success of mobile computing, especially the limited battery life restricts the use of smartphones in business and personal applications which require intensive computation and/or long running time [3].

A mobile cloud workflow in this article is an application that can be recognized as a set of tasks in a partial order to achieve a specific business goal (e.g. an order processing workflow in an online sale system, and a travel approval workflow in a mobile office system) using a group of mobile devices and cloud servers [10]. Generally speaking, all the tasks can be executed either on the mobile locally or on the cloud servers remotely in a specific order determined by the workflow logic. Different from the traditional mobile

The associate editor coordinating the review of this manuscript and approving it for publication was Huan Zhou.

application, a mobile cloud workflow has three main features (here we name the mobile user who triggers the workflow as ‘‘client’’, and name the resource which executes the workflow task as ‘‘processing unit’’):

- The client and the processing unit are interacting with each other. In MCC, the client can trigger the workflow and control the interaction with the processing unit. For example, the client may request the workflow state to roll back from the current GUI (Graphical User Interface) state to a preceding state if required.
- The processing unit can be either the mobile device or the cloud server. Generally speaking, in MCC, a workflow task can be executed by either the mobile device locally or the cloud server via computation offloading. Therefore, a cloud server can not only act as a required service for workflow execution, but also play a key role to receive and handle the requests for computation offloading sent from the mobile device.
- Human intervention is normal. In MCC, a mobile cloud workflow often cannot be executed in a fully automatic fashion as human intervention is needed to complete some workflow tasks. That is, the execution of the mobile cloud workflow may involve not only the ‘‘client’’ and the ‘‘processing unit’’, but also some other human participants.

These features of mobile cloud workflow can seriously increase the complexity in energy-efficient resource allocation due to the following reasons:

1) MORE UNCERTAINTIES DURING THE WORKFLOW EXECUTION

There are increasingly more uncertain factors during the execution of the mobile workflow, for example, the user may be away from the mobile while the workflow is waiting for the input, and the user may restart a new workflow from the mobile side without stopping the old tasks running on the cloud and so on. Meanwhile, there may be other users involved to deal with certain workflow tasks. Thus, these user interventions will inevitably increase the waiting time of individual workflow task and further increase its execution time, which will lead to the increase the uncertainty and the risk of task time delay. If there are too many user interventions in an active workflow, too many individual workflow tasks may be delayed which will increase the risk of the workflow execution time. To deal with these risks, deadlines on individual workflow tasks are normally assigned to ensure the timely completion of the whole workflow. Therefore, besides workflow deadlines, we also need to consider task deadlines in our energy-efficient resource allocation strategy.

2) LOCATION RESTRICTIONS

Some workflow tasks can only be executed on the cloud or the mobile device as the execution requires specific data, software or hardware that are only available at a specific location due to technology limitation or security concerns. Therefore, we need to consider these location restrictions when design the energy-efficient resource allocation strategy.

3) CONFILICT BETWEEN SAVING ENERGY AND IMPROVING QOS

The processing power of the cloud server is usually higher than the mobile device so that QoS (Quality of Service) especially execution time could be improved when tasks are executed on the cloud. However, offloading the workflow tasks from the mobile onto the cloud will consume additional time, network and energy. Therefore, there can be a conflict between saving the energy and improving the QoS, and such a conflict needs to be addressed in our energy-efficient resource allocation strategy.

To tackle these challenges above, we first investigate the execution features of a real-world mobile cloud workflow and determine some relevant settings. For the sake of maximizing the benefits of MCC in energy saving and service quality (specifically on execution time) improvement, we synthesize the energy consumption and execution time as a type of UtilityCost of the mobile device and then formulate the resource allocation problem as an optimization problem with the goal of minimizing the UtilityCost while meeting the workflow deadline constraints. Furthermore, to address this problem, we propose an energy-efficient resource allocation algorithm shortened as **EERA** to solve the problem. EERA firstly finds all the Partial Critical Paths (PCP) of the workflow graph to segment the whole optimization problem into several local problems, and then uses the Discrete Particle Swarm Optimization (DPSO) to find the local optimal solution for every local problem, finally the local optimal solution is integrated to obtain the global one. In addition, we adapt, implement and compare several representative algorithms including Greedy algorithm (which has been further implemented into two versions: Greedy on UtilityCost (**GU**) and Greedy on Energy (GE)), Heterogeneous Earliest Finish Time (HEFT), Genetic Algorithm (GA) and DPSO to make the resource allocation decision. To evaluate the performance of our resource allocation strategy, simulation experiments are conducted and comparison results on three key measurements including energy consumption of the mobile device (ECMD), execution time of the workflow (ET/WD) and the UtilityCost (UC), are presented and discussed. The results show that our proposed strategy EERA achieves the best overall performance as it significantly reduces the energy consumption while achieving the best trade-off between the energy consumption and execution time.

To sum up, the major contributions of our work are summarized in three aspects:

- We task, design and analyze an online travel planning workflow as a typical example to demonstrate the special features of the mobile cloud workflows and investigate the energy-efficient resource allocation strategy for them which are verified more complicated in the real-world for achieving business goals. Based on this, we are more aware of that an energy-efficient resource allocation strategy is of vital importance in MCC system.

- Comprehensive models, including workflow application model, MCC system model, execution time model, power consumption model and resource allocation optimization model, are designed for the problem and especially a novel concept of "UtilityCost" has been proposed to balance the energy consumption of the mobile device and the total execution time of workflows. To the best of our knowledge, the proposed optimization model with minimizing the UtilityCost of the system is the first work jointly and evenly the energy consumption and the execution time.
- Furthermore, an energy-efficient resource allocation algorithm shortened as EERA, combining the PCP and DPSO algorithm, has been proposed to solve the optimization problem. In addition, several heuristics/metaheuristics-based representative algorithms have been designed and adopted to verify the effectiveness and superiority of EERA algorithm. Finally, comprehensive experiments with various parameters and different graph structures have been conducted and implemented to verify its effectiveness and the simulation results illustrate that EERA outperforms other representative algorithms with three key measurements, including the ECMD, ET/WD and UC.

The remainder of this article is organized as follows: Section II presents a motivating example and the problem analysis. Section III defines all related models used in this article. Section IV proposes our energy-efficient resource allocation strategy as well as three other presentative strategies. Section V demonstrates the evaluation results. Section VI presents the related work. Finally, Section VII concludes the paper and points out some future work.

II. MOTIVATION EXAMPLE AND PROBLEM ANALYSIS

In this section, we illustrate a representative motivating scenario for a real-world mobile cloud workflow. Afterwards, the workflow features mentioned in the introduction section have been further analyzed using this example.

A. AN ONLINE TRAVEL PLANNING WORKFLOW

An online travel planning workflow is a typical mobile cloud workflow which involves not only two representative roles including "client" and "service provider", but also the third-party participator such as "financial auditor". This workflow will be accomplished through several tasks executed either locally or remotely in a defined order and within limited time. The flowchart in Fig.1 illustrates the basic workflow process after a mobile user request arrives into the travel planning application.

There are a large number of mobile clients using this online travel planning application at the same time every day and the first task is to log into the application when a travel planning request arrives. In fact, among these mobile clients

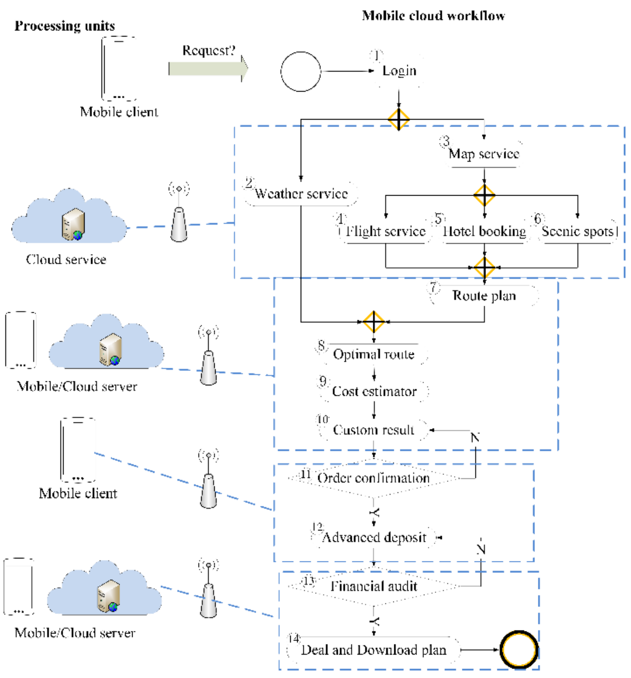


FIGURE 1. Online travel planning workflow flowchart.

some may have logged into the application recently, but some may be new users or users which have logged out of the application for a long time. Accordingly, the login service can be executed on the mobile devices locally or on the cloud servers remotely. When the client login is successful, he/she must input the basic information about the travel, such as number of people, destination, travel dates and other user preference. Upon receiving the information, the weather/map service first generates the local map and relevant seasonal hint (Task 2, 3). With the help of a local map, the flight service can find the proper ticket information, the hotel booking service can search and return the hotel information and other relevant information such as local scenic spots which are obtained by Task 6. Some possible route plans are generated by synthesizing the local weather, map, flight, hotel and scenic spots (Task 7). According to the user preference, an optimal travelling plan is selected by a specific recommendation algorithm (Task 8). Following that, the cost for the travel can be estimated and the travel planning result is returned and displayed to the mobile client (Task 9, 10). Afterwards, the client affirms if he/she is satisfied with the travel planning result (Task 11). If yes, then the client needs to pay an advanced deposit which will be carefully checked by the financial auditor (Task 12, 13). Finally, the mobile client can download the travel plan and the receipt for the deposit, and the whole workflow is accomplished (Task 14).

B. PROBLEM ANALYSIS

In the online travel planning workflow example, each task has its own features during its execution: for example, some of them are computation-intensive tasks (Task 7 and 8), some of them are data-intensive tasks (Task 14), and some others are communication-intensive tasks (Task 11 and 13). Except for some special tasks, most tasks can be executed either on the mobile locally or offloaded onto the cloud server remotely. We denote these tasks without location constraint for execution as free nodes and those special tasks with location constraint as fixed nodes. For example, to get the local map and local weather information, the tasks must access to the cloud services (Task 2, 3) which must be executed on the cloud remotely. But the other tasks such as login the application (Task 1) and confirm the order (Task 11) can only be executed on the mobile device locally by the end user. Thus, in this example workflow, Tasks 2-6 are ‘‘fixed’’ nodes and others are ‘‘free’’ nodes. Meanwhile, Task 13 needs a third participant ‘‘Financial Auditor’’ to check whether the advanced deposit paid by the client meet the minimum cost or not, which belongs to the human intervention.

Based on the above analysis and to further obtain some quantitative results, we implement the travel planning workflow application using the Android SDK in JAVA. We set the tasks (Task 2-6) as the fixed nodes which can be executed only on the cloud, and the task ‘‘order confirmation’’ (Task 11) as fixed node which can be executed only on the mobile device. Except for these nodes, other nodes are free nodes which can be executed either locally or remotely. In addition, a Huawei Honor 5C mobile phone with Android 6.0 is used as the testing mobile device and a desktop PC with Windows 10 which has dual-core CPU is used as testing cloud server. We also use PowerTutor [11] to monitor the energy usage of the major mobile system components. We consider two situations for comparison purpose: one is that all free nodes are executed on the mobile device locally and the other one is that all free nodes are executed on the cloud server remotely.

As shown in Fig.2, when the travel time is short (lower than 7 days), offloading the free nodes onto the cloud for execution consumes less energy than processing them on the mobile device locally. But when the travel time becomes longer (more than 7 days), the mobile device will consume much more energy for task offloading, and on the contrary, the device will consume less energy when executing the free nodes locally. This is due to a significant increase in the energy consumption and network usage for the data communication between the mobile device and the cloud.

As for the workflow execution time, it is obvious that offloading the free nodes onto the cloud is more time-consuming than running them on the mobile device locally in all situations with different travel times as demonstrated in Fig.3. On average, executing all tasks remotely spends 2.67ms longer than executing all tasks locally, which however can be considered as negligible.

Form the comparison results above, we can see that offloading the free nodes onto the cloud for execution consumes both extra time and energy. If the whole execution time of the online travel planning workflow is within a normal range and the travel time is short, offloading the free nodes can reduce the energy consumption of the device. In contrast, if the travel time is longer, executing all the nodes locally

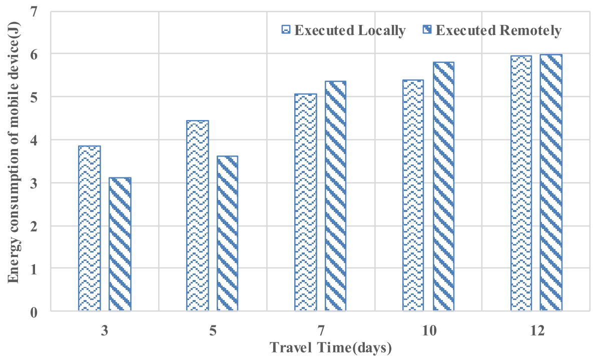


FIGURE 2. Energy consumption of the mobile device.

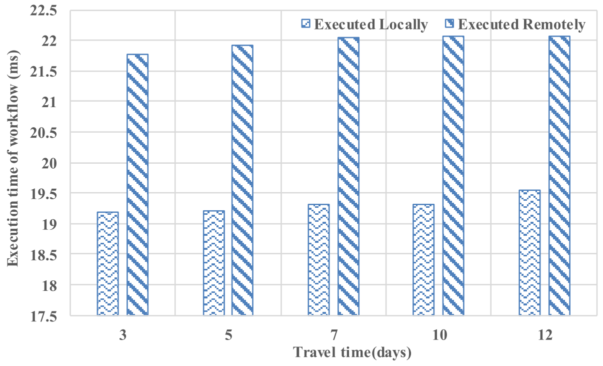


FIGURE 3. Execution time of the online workflow.

can obtain a lower energy consumption. Therefore, we need to synthesize the features of the workflow tasks, as well as the network environment and the QoS constraints, in order to design an efficient resource allocation strategy which can achieve the minimum energy consumption while meeting the execution time constraints.

III. RESOURCE ALLOCATION MODELS

In this section, we will describe the resource allocation models for mobile cloud workflows in detail.

A. WORKFLOW APPLICATION MODEL

A mobile cloud workflow in MCC environment is a process that consists of a series of workflow tasks performed in series or in parallel to be automated and executed collaboratively. An important feature of mobile cloud workflow is that the mobile client can request the workflow to transition from the current state to a preceding state. Thus, the dependency relationship ‘‘loop’’ may happen in the execution of the workflow. For this situation, we convert the loop to a sequence by adding a middle task illustrated. And through this conversion, the workflow with cycles can be transformed to be acyclic that can be modeled as a Directed Acyclic Graph.

Definition 1 (Mobile Cloud Task): A mobile cloud task in MCC is a relatively independent computation unit which is invoked inside a workflow and can be executed in a mobile device or cloud server. Each task receives an input and produces an output after processing. We model a mobile task as a triple v=(dI, dO,w) where dI/dO is the size of the input/output data, and w is the workload size of the task for processing.

Definition 2 (Mobile Cloud Workflow): A mobile cloud workflow is a process that consists of a series of mobile cloud tasks executed in a specific order with a goal of accomplish the mobile user's request. We use a Directed Acyclic Graph G=(V, E) to model a mobile cloud workflow with n tasks, where V={v_1, v_2, ... v_i, ... v_n} represents a set of mobile cloud tasks and E={$e(v_i, v_j)|v_i, v_j \in V$} is illustrates the invoke relationship between task v_i and v_j with the constraint that the start execution time of task v_j must be later than the finish time of task v_i.

B. MCC SYSTEM MODEL

Definition 3 (Mobile Device): A mobile device is a smartphone, tablet or any other portable device which has access to the Internet and can invoke resources from the cloud. We model the mobile device as a 4-tuple MD=(f_M, P_0, P_{up}, P_{down}) where f_M is the operating frequency of the mobile device for executing a task, P_0 is the idle power consumption under the mobile device startup state, P_{up} and P_{down} denote the power for uploading and downloading data of the mobile device, where P_{up} is considerably larger than P_{down}. They are all fixed value in the same MCC environment and can be measured based on experience. It is well known that the power consumption of mobile P_M is proportional to its operating voltage squared and frequency, and the operating voltage of the core is proportional to frequency. Thus, the power consumption can be represented as $P_M = \alpha f_M^{\gamma}$, where both α and γ are constants related to the mobile device and can be obtained through testing.

Definition 4 (Cloud Server): A cloud server in MCC is a computing element in the cloud which can provide service to the mobile devices or execute the computation offloading from the mobile devices. We use f_C to represent the operating frequency of cloud server.

Definition 5 (Resource Allocation Decision): A resource allocation decision is a vector for each mobile cloud workflow to determine which task is running locally and which task is executed remotely. We introduce a 0-1 variable x_i to denotes the decision of offloading for task v_i, specifically $x_i = 0$ represents that the task will be executed locally on the mobile device and $x_i = 1$ represents that the task will be offloaded to the cloud server for execution. The resource allocation decision can be denoted as the set $X = \{x_1, x_2, \ldots x_i, \ldots, x_n\}$.

C. EXECUTION TIME MODEL

The workflow execution time contains two parts: the computation time of each task and the communication time of each arc. The computation time of task v_i depends on its execution decision. And the communication time of each arc e_{ij} depends on the decisions of task v_i and v_j. Thus, we denote $T_i^{cp}(x_i)$ as the computation time of task v_i and $T_{ij}^{cm}(x_i, x_j)$ as the communication time between tasks v_i and v_j, where x_i and x_j represent the allocation decision of v_i and v_j, respectively. We formulate the task execution time as follows:

1) COMPUTATION TIME

When task v_i is executed on the mobile, $x_i = 0$, the execution time of task x_i can be calculated by $T_i^{cp}(x_i = 0) = w_i/f_M$. If execute the task v_i on the cloud, its computation time can be calculated by $T_i^{cp}(x_i = 1) = w_i/f_C$, illustrated in (1).

$$T_i^{cp}(x_i) = \begin{cases} w_i/f_M, & x_i = 0 \\ w_i/f_C, & x_i = 1 \end{cases} \tag{1}$$

2) COMMUNICATION TIME

If the decision of task v_i and task v_j is equal, both executed on the cloud/mobile, the communication time is 0; if task v_i is executed on the mobile and its immediate successor is executed on the cloud, the communication time is $Sdata_{ij}/R_{up}$ where $Sdata_{ij}$ is sending data from mobile to cloud and R_{up} is the uploading rate; and if task v_i is executed on the cloud and its immediate successor is executed on the mobile, the communication time is $Rdata_{ij}/R_{down}$ where $Rdata_{ij}$ is the receiving data from the cloud to mobile and R_{down} is the downloading rata; To summaries, the communication time between tasks v_i and v_j can be calculated by (2).

$$T_{ij}^{cm}(x_i, x_j) = \begin{cases} 0, & x_i = 0, x_j = 0 \\ Sdata_{ij}/R_{up}, & x_i = 0, x_j = 1 \\ Rdata_{ij}/R_{down}, & x_i = 1, x_j = 0 \\ 0, & x_i = 1, x_j = 1 \end{cases} \tag{2}$$

Combining the above analysis, the execution time of application can be calculated by (3), where T_{n+1}^{finish} is the finish time of the last task in the workflow, T_{n+1}^{cp} is the computation time of the last task in the workflow, and $T_{i,n+1}^{cm}$ is the communication time between the last task v_{n+1} and its predecessor task v_i.

$$T(X) = T_{n+1}^{finish} = T_{n+1}^{cp}(x_{n+1}) + T_{i,n+1}^{cm}(x_i, x_{n+1}) \tag{3}$$

D. POWER CONSUMPTION MODEL

Power consumption of the mobile device also contains two parts: computation power consumption of the tasks and communication power consumption of the arcs, which is affected by the resource allocation decision. We use $E_i^{cp}(x_i)$ to denote the power consumption during the execution time of task v_i with x_i. $E_{ij}^{cm}(x_i, x_j)$ denotes the power consumption of arc e_{ij} with x_i and x_j.

1) COMPUTATION POWER CONSUMPTION

When task v_i is executed locally, the power consumption only contains the computation power consumed on the device, calculated by $\mathrm{E_i^{cp}(x_i = 0) = P_M * T_i^{cp}(x_i = 0)} = \alpha \mathrm{f_M^{\gamma-1}}$. If execute the task v_i remotely, the power consumption of mobile $E_i^{cp}(x_i = 1) = P_0 * T_i^{cp}(x_i = 1) = P_0 * w_i/f_C$.

$$E_i^{cp}(x_i) = \begin{cases} \alpha f_M^{\gamma-1}, & x_i = 0 \\ P_0 w_i/f_C, & x_i = 1 \end{cases} \tag{4}$$

2) COMMUNICATION POWER CONSUMPTION

The power consumption during the communication process depends on the resource allocation decisions of two tasks in the arc e_{ij}. If task v_i is executed on the cloud/mobile and its immediate successor v_j is executed on the same resource, the consumed communication power is regarded as 0; but if task v_i is executed on the mobile and task v_j is executed on the cloud, the consumed communication power is calculated as $E_{ij}^{cm}(x_i = 0, x_j = 1) = P_{up}*Sdata_{ij}/R_{up}\mathrm{Tdata_i/f_s}$; if task v_i is executed on the cloud and task v_j is executed on the mobile, the consumed communication power is calculated as $E_{ij}^{cm}(x_i = 1, x_j = 0) = P_{down}*Rdata_{ij}/R_{down}\mathrm{Rdata_i/f_r}$; To summaries, the communication power consumption of two connective tasks v_i and v_j with the offloading decision x_i and x_j can be calculated by (5) as follows.

$$E_{ij}^{cm}(x_i, x_j) = \begin{cases} 0, & x_i = 0,\ x_j = 0 \\ P_{up}Sdata_{ij}/R_{up}, & x_i = 0,\ x_j = 1 \\ P_{down}Rdata_{ij}/R_{down}, & x_i = 1,\ x_j = 0 \\ 0, & x_i = 1,\ x_j = 1 \end{cases} \tag{5}$$

Based on the calculated components mentioned above, the energy consumption of the mobile in MCC can be denoted as (6).

$$E(X) = \sum\nolimits_{i,j\in V} \left(E_i^{cp}(x_i) + E_{ij}^{cm}\left(x_i, x_j\right)\right) \tag{6}$$

E. RESOURCE ALLOCATION OPTIMIZATION MODEL

Since the goal of minimizing the total energy consumption and goal of minimizing the total execution time are contradictive to each other to some extent, the target is to find the best trade-off between them. For such a purpose, we define the tradeoff as the UtilityCost of the decision as in (7), where $E(X)$ and $T(X)$ represent the total energy consumption of the mobile device and the total execution time of the workflow respectively with the resource allocation decision X. The denominator in (7) represents the total energy consumption of the mobile $E(X_0)$ and the execution time of the workflow application $T(X_0)$, when all the free tasks are executed locally besides some fixed tasks, that is, $X = X_0 = \{0, 0, 0 \ldots\}$. Because that there are no communication time when all the tasks are executed locally, thus $E(X_0)$ can be calculated as $\sum_{k=0}^{n+1} E_k^{cp}(X_0)$ and $T(X_0)$ can be calculated as $T_{n+1}^{cp}(x_{n+1})$ according to (3) and (6).

$$U(X) = \frac{E(X)}{E(X_0)} * \frac{T(X)}{T(X_0)} = \frac{E(X)}{\sum_{k=0}^{n+1} E_k^{cp}(X_0)} * \frac{T(X)}{T_{n+1}^{cp}(x_{n+1})} \tag{7}$$

Meanwhile, there are some constraints of the optimization problem.

Constraint 1 (Whole Execution Time): the whole execution time of the workflow should be less than the execution deadline T_{MAX} denoted as (13);

Constraint 2 (Partial Execution Time): the execution of mobile cloud workflow involves human intervention, which leads to a big uncertainty of time. Hence, we must make some constraints in partial execution time, that is, tasks should be completed in a required time, denoted as (9);

Constraint 3 (Fixed Tasks): in fact, there are some mobile cloud tasks only can be executed on the cloud or on the device, which have no choices to choose the other way. We defined these special tasks as fixed tasks and divide them into two categories: set M denotes the tasks only can be executed on the mobile device and set C denotes the tasks only can be executed on the cloud serve, illustrated as (10);

Therefore, the optimization resource allocation problem can be further formulated as a minimization problem of the UtilityCost with constraints (7)-(12). The objective is to find the optimal offloading decision X that can minimize the UtilityCost in (7). The explanation of constrains (8)-(10) is abovementioned, the constraint in (11) guarantees that the offloading decision is a 0-1 variable and the constraint in (12) makes sure that two dummy tasks must be executed on the mobile locally.

$$\text{Min } U(X) = \frac{E(X)}{\sum_{k=0}^{n+1} E_k^{cp}(X_0)} * \frac{T(X)}{T_{n+1}^{cp}(x_{n+1})} \tag{8}$$

$$\text{s.t. } T(X) \le T_{MAX}$$

$$T_i^{cp}(x_i) + max_{e_{ij}\in E} T_{ij}^{cm}(x_i, x_j) \le T, max_i\ (i, j \in [1, n]) \tag{9}$$

$$\text{For each } v_i \in M, x_i = 0;\ v_i \in C, x_i = 1 \tag{10}$$

$$x_i \in [0, 1](x_i \in X, i \in [1, n]) \tag{11}$$

$$x_0 = x_{n+1} = 0 \tag{12}$$

IV. GREEN RESOURCE ALLOCATION STRATEGY AND CANDIDATE ALGORITHMS

Green resource allocation strategy is to optimize the task offloading decision X in the mobile cloud environment. The major goal for the green resource allocation strategy is to minimize the UtilityCost of the mobile cloud workflow while all the QoS constraints can be satisfied by dynamically optimizing the decision X. Note that given the large scalability of the mobile cloud workflow, there will be a huge number of tasks running in the system. Further, The resource allocation problem of minimizing the UtilityCost with constraints is an NP problem [33], [34], and hence it is difficult to design a global algorithm to find the optimal solution within a reasonable overhead. Therefore, in this article, we propose an improved DPSO based algorithm named Energy-Efficient Resource Allocation (shortened as **EERA**) algorithm based on Partial Critical Path (PCP) to solve the optimization problem.

A. ENERGY-EFFICIENT RESOURCE ALLOCATION ALGORITHM

The core idea of EERA is that we start from the exit task of the workflow, find all the partial critical paths (PCPs) recursively, and then use the DPSO algorithm with the goal of minimizing the UtilityCost to schedule all the free nodes on each PCP.

Critical path, the longest execution path between the entry and exit tasks of a workflow is widely used in task graph.

The key of designing an optimal X focuses on the decisions of the tasks on the critical path. In this article, we use the PCP to distribute the overall deadline of the workflow into several sub processes and assign each critical task on the PCP recursively until all the tasks assigned [17], [19].

1) BASIC DEFINITION

In order to find all the PCPs in G, we define some basic notions: the earliest start time $EST(v_i)$ denoted as the earliest start time of task v_i for which it will start the execution and the latest finish time $LFT(v_i)$ denoted as the latest finish time of task v_i for which it has finished the execution. The task computation time and communication time may be different with different decision, which has a great effect on obtaining the exact earliest start time $EST(v_i)$ and the latest finish time $LFT(v_i)$. Thus, we use an approximate value to compute the $EST(v_i)$ and $LFT(v_i)$ of each task. Assuming that all the tasks are executed on the cloud with f_C, we can compute the earliest start time $EST(v_i)$ by (13).

$$\begin{cases} EST(v_0) = 0 \\ EST(v_i) = \max_{v_j \in P(v_i)}\{EST(v_j) + T_j^{cp} + T_{ji}^{cm}\} \end{cases} \tag{13}$$

where the set $P(v_i)$ represents the parent task set of v_i. And similarly, we define the latest finish time $LFT(v_i)$ as formula (14), where $C(v_i)$ is the children tasks set of v_i.

$$\begin{cases} LFT(v_{n+1}) = T_{max} \\ LFT(v_i) = \min_{v_j \in C(v_i)}\{LFT(v_j) - T_j^{cp} - T_{ij}^{cm}\} \end{cases} \tag{14}$$

2) GREEN RESOURCE ALLOCATION ALGORITHM BASED ON PCP

Algorithm 1 illustrates the algorithm based on PCP. In line 2, we add two dummy nodes v_0 and v_{n+1} in the task execution graph, which have no actual scheduling in the algorithm. Line 3 and line 4 give the computation process of earliest start time and latest finish time for each task in G. Line 5 indicates that the entrance task and exit task must be executed on the mobile and set as scheduled nodes. We define the scheduled node as a task that has made its decision for execution and when all the tasks in G have been scheduled, the strategy terminates. In the end, we call the PCP() procedure to find the PCPs.

Algorithm 1 EERA (G, T_{max})

Input: Workflow: G and time deadline T_{max}
Output: the optimal decision X^*

```
1: Procedure EERA(G, T_max)
2:     Add v_0, v_{n+1} and their corresponding edges to G;
3:     Compute EST for each task by (13);
4:     Compute LFT for each task by (14);
5:     Set task v_0 and v_{n+1} are scheduled;
6:     Call PCP(v_{n+1}).
```

3) FINDING THE PARTIAL CRITICAL PATH

We can find the PCP for a given task v by **Algorithm 2** and the whole PCPs will be found recursively. First of all,

Algorithm 2 PCP (v)

Input: the task v
Output: the offloading decision X^*

```
 1: Procedure PCP(v)
 2: if (v has no unscheduled parent) then
 3:     return(Success);
 4: PCP←empty;
 5: u ← v;
 6: While ( u has an unscheduled parent) do
 7:     Find the critical parent p(u) unscheduled;
 8:     Add p(u) into the beginning of PCP;
 9:     u ← p(u);
10: if (PCP is not scheduled) then
11:     Call DPSO(PCP);
12: For all( task v′ on PCP) do
13:     if (v′ has an unscheduled parent) then
14:         Call PCP( v′);
15:         Compute EST for each task on PCP by (13);
16:         Compute LFT for each task on PCP by (14);
```

if the task v has no unscheduled parent, the offloading has been finished (Line 2-3). Then set the PCP be empty(Line 4). And in the first while loop (from Line 6-9), the PCP of a particular task will be constructed by finding its critical parent task unscheduled. In the basic Graph Theory, the parent task that results in the maximum earliest start time is the critical parent of the task. After that, we add the scheduled critical parent as the beginning of PCP. Repeat the process several times until the task has no unscheduled parent. Followed that, the algorithm calls the procedure DPSO(PCP) (illustrated in Section B) with the input variable, PCP, to make the task execution decision and schedule the tasks on the PCP before its latest finish time (Line 10-11). After this procedure, the earliest start time and latest finish time of each task on the PCP need to be updated and call the procedure recursively to schedule all the tasks in the workflow graph (Line 12-15).

B. DISCRETE PARTICLE SWARM OPTIMISATION

Particle Swarm Optimization (PSO) is an evolutionary algorithm proposed by J. Kennedy and R. C. Eberhart which is designed to find the optimal solution for continuous optimization problems. In this article, we introduce a discrete version, named DPSO, to solve our resource allocation problem [15], [16]. In the resource allocation model, the one-dimensional task execution decision X in G can be represented as a particle position in DPSO. Each particle adjusts its flying velocity according to its flying experience (local best position: $pbest_i$) and the companion's experience (global best position: $gbest$). Thus, the position of particle i can be denoted as $X_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{in})$, $1 \le j \le n$, $x_{ij} \in [0,1]$, a particle i is associated with a n-dimensional velocity $V_i = (v_{i1}, v_{i2}, \ldots, v_{in})$, $v_{ij} \in [-1,0,1]$ to determine its flying distance and direction. $v_{ij} = -1$ denotes the particle i is learning from $pbest_i$, $v_{ij} = 0$ denotes the particle i is learning from itself and $v_{ij} = -1$ denotes the particle i is learning from *gbest*. We define the velocity by $V = V_p + V_g$, where V_p denotes the velocity adjusting part towards its local best position *pbest*, and V_g denotes the velocity adjusting part towards the global best position *gbest*. The calculation rules of these variables in our DPSO algorithm is illustrated by the formulas as follows:

$$V_p = c_1(X_{pbest} - X)$$
$$V_p = \begin{cases} 0, & rr < pp \\ 1, & otherwise \end{cases} \tag{15}$$

In (13), c_1 represents the learning parameter towards *pbest*, and we generate a random indicative parameter *rr* for each particle, define *pp*∈[0,1] as a learning threshold towards *pbest*, obtain V_p by comparing *rr* and *pp*. When $rr < pp$, $V_p = 0$ indicates that the new particle will select the corresponding position from itself, otherwise, $V_p = 1$ indicates that the new particle will select the corresponding position from its local best position.

In a similar way, V_g can be defined in (16), where c_1 represents the learning parameter towards *gbest*. *gp*∈[0,1] is a learning threshold towards *gbest*, *rr* is a random indicative parameter introduced above, and V_g can be calculated by comparing the value of *rr* and *pg*. When $rr < gp$, V_g =0 indicates that the new particle will select the corresponding position from itself, otherwise, $V_g = 1$ indicates that the new particle will select the corresponding position from the global best position.

$$V_g = c_2(X_{gbest} - X)$$
$$V_g = \begin{cases} 0, & rr < gp \\ -1, & otherwise \end{cases} \tag{16}$$

Thus, the velocity of each particle can be calculated by: $V = V_p + V_g = c_1(X_{pbest}\text{-}X) + c_2(X_{gbest}\text{-}X)$, and the new position of each particle can be updated by $X = X + V$.

Algorithm 3 DPSO(G) illustrates the pseudocode of our DPSO based heuristic resource allocation algorithm in G. As depicted in DPSO(G), before the DPSO starts, we need remove the fixed tasks from PCP and obtain a new path G' (Line 1-2). The objective of the searching phase here is to find the global optimal resource allocation strategy for G' using DPSO (Line 3-15). The algorithm starts with the initialization of the swarm (Line 4). After generating initial feasible position X for *MS* particles, *pbest* can be initialized. Then the initial *gbest* with minimum Utility can be obtained (Line 5). The learning probability pp for learning *pbest* and *gp* for learning *gbest* (Line 6) are generated randomly. The evolution phase of DPSO is Line 7-15 and it comes to the end until the stop condition: the maximum iteration number. For each particle, firstly generate a random indicative parameter *rr*, then calculate V_p, V_g through comparing *rr* with *pp*, *gp*, and finally calculate the velocity V using V_p and V_g (Line 10-12). After that, a new swarm can be obtained by $X_i = X_{i-1} + V$. Next step is to check the feasibility of X_i with all the

Algorithm 3 DPSO (G)

```
Input: Workflow: $G$; Fixed task set $M, C$;
Maximum iteration: $N^{max}$; Population size: $MS$.
Output: Task offloading decision on PCP: $X^*$
1:  for each task in $G$ do
2:      {Remove the fixed tasks in $M$ and $C$ from $G$ and
        obtain $G$'}
        //Swarm initialisation
3:  For $i = 1$ to $MS$ do
4:      {Generate a feasible solution $X$ and $pbest$=$X$;} //
    $pbest$ equals to the solution itself initially.
5:  Find $gbest$;
6:  Generate the learning threshold $pp$ and $gp$ randomly;
    //generate the new swarm
7:  While ($j <= N^{max}$) do {
8:      For $i = 1$ to $MS$ do{
9:          Generate a random indicative parameter $rr$;
10:         Calculate $V_p$ and $V_q$;
11:         Calculate $V$ using $V_p$ and $V_g$;
12:         Calculate the new position $X_j$; //
13:         Feasibility($X_i$);
14:         Updata $pbest$ and $gbest$;}
15: Return $X^*$=$gbest$;}
```

constraints (Line 13). Following that, update *pbest* and *gbest* for the swarm (Line 14). The last step of the whole algorithm is return the global best position of the particle and deploy it to X^*.

C. COMPLEXITY ANALYSIS OF EERA ALGORITHM

The proposed EERA algorithm is depend on PCP and DPSO. The complexity of PCP algorithm is $O(n + e)$, where n is the number of vertices and e is the number of edges in directed acyclic graph G. And the complexity of DPSO algorithm is $O(N^{max}*MS)$, where N^{max} is its maximum iteration and MS is the population size of the particle. In the worst case, all the possible task offloading decision can be a particle. The maximum population size is 2^m, where m is the number of the tasks in the *PCP*. And m is much less than n due to the PCP algorithm. In addition, the *PCP*s in the workflow graph is becoming shorter and shorter during the recursive lookup, which leads to the value of N^{max} and MS smaller and smaller.

Thus, the complexity of *EERA* algorithm is max{$O(n + e)$, $O(N^{max}*MS)$}.

V. EXPERIMENTAL EVALUATION

In this section, we first introduce other representative task scheduling algorithms including Greedy (GU and GE), HEFT, GA and DPSO briefly. Then, experimental settings, including the experimental computing environment, the parameter settings of workflows, resources and algorithms, are illustrated. Finally, we illustrate the simulation results to evaluate the performance of our proposed resource allocation strategy EERA and demonstrate the comparison results with other algorithms.

A. INTRODUCTION OF ORTHER REPRESENTATIVE ALGORITHMS

For the comparison purpose, we employ and adapt several representative heuristics or metaheuristics-based resource allocation algorithms including Greedy, Heterogeneous Earliest Finish Time, Genetic Algorithm, and Discrete Particle Swarm Optimization (DPSO) introduced in the above part as the candidate resource allocation algorithms.

Greedy algorithm is one of the simplest heuristic algorithms to solve the optimization and search problems [12]. In Greedy, every solution is a local best choice in the current iteration. That is, the optimal solution obtained by Greedy is just a local optimal one, but not a global one. The key issue of Greedy is the choice of the greedy strategy. For comparison, we introduce two different Greedy algorithm versions according to the greedy strategy. One version uses a greedy strategy with minimum UtilityCost, named as Greedy on UtilityCost (GU) and the other one uses a greedy strategy with minimum energy consumption of the mobile device, named as Greedy on Energy (GE).

The Heterogeneous earliest finish time (HEFT) proposed by Topcuouglu *et al.* is a two-stage static algorithm, including task prioritization and resource selection, to solve the optimization resource allocation problem [13].

Genetic Algorithms (GA) is one of the most commonly used search techniques to generate high-quality or approximate solutions for the optimization and search problems [14]. GA is inspired by the process of natural selection to make the evolution toward better solution by evolutionary biology. In GA, every solution is represented with a string, also known as a chromosome or an individual which is defined by the encoding method. After encoding, the initial population with M individuals need to be generated as the initial search space. Within each generation/iteration, a new population can be obtained by using three basic GA operations, i.e. Gene selection, Gene crossover, and Gene mutation, to simulate the gene evolution process. Finally, the individual with the best fitness value is returned to represent the best optimal solution when meet the terminated condition. Thus, the whole GA process is end. In recent years, GA has been considered as an effective method to deal with the resource allocation problems.

Besides the three algorithms introduced above, we also implement other popular algorithms including Local which executes all free tasks on the mobile device locally and DPSO introduced in Part IV(B) for comparison.

B. EXPERIMENTAL SETTINGS

1) EXPERIMENTAL ENVIRONMENT SETUP

A PC having an Intel Core i5 processor with a maximum frequency of 4.2 GHz and a 16G RAM, which installed a Windows 10 system, is used to test our proposed resource allocation strategy. We used the MATLAB 2015 integrated development environment to generate the workflows with different scale, implement the proposed EERA algorithm and other comparison algorithms.

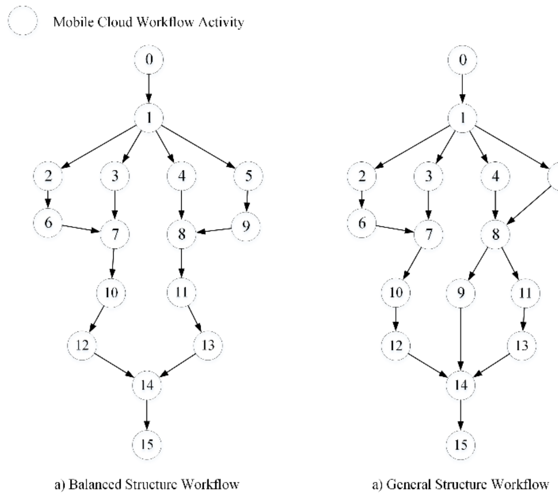

Mobile Cloud Workflow Activity
a) Balanced Structure Workflow
a) General Structure Workflow

FIGURE 4. Two different structures for workflows.

2) PARAMETER SETTINGS FOR MOBILE CLOUD WORKFLOWS

The workflow process depends on its structure, task workload and the I/O data. Specifically, according to the task execution logic in the workflow, we consider two different graph structures: balanced and general in this article as shown in Fig.4. The former illustrated as Fig.4(a) refers to the workflows in that the tasks are executed in parallel and symmetric. And the latter one illustrated as Fig.4(b) refers to the workflows in that the tasks are executed irregularly and generally. Additionally, the scale of the workflow changes from 12 to 102 tasks (including the two dummy tasks), each workflow has 10% tasks set as fixed-position tasks, and the arcs number in the graph is set as double times the tasks. On average, we divide the tasks in a workflow into two different types: one is computation-intensive and one is communication-intensive. For the computation-intensive tasks, the mean workload of each task is randomly selected from 5000 to 8000 and its standard deviation is defined as 33% of its mean. The mean sending/receiving data of each arc is randomly selected from 50 to 3000 and its standard deviation is defined as 33% of its mean. To the contrary, for the communication-intensive tasks, the mean workload of each task is randomly selected from 50 to 3000 and its standard deviation is defined as 33% of its mean. The mean sending/receiving data of each arc is randomly selected from 5000 to 8000 and its standard deviation is defined as 33% of its mean.

3) PARAMETER SETTINGS FOR CONSTRAINTS AND RESOURCES

It is well known that the execution deadline has a major impact on the performance of the resource allocation strategies. In order to evaluate all the strategies under a reasonable workflow deadline constraint, we specify the workflow deadline using (17), where T_{max} and T_{min} are the workflow execution time produced by GE and HEFT respectively. We define k as a relaxation factor between 0 and 10 to evaluate the impact of the deadline constraint from tight to relax. The larger k is, the more relax the constraint is.

$$WD = T_{min} + k(T_{max} - T_{min}) \tag{17}$$

Same as the mobile cloud environment for the motivating example discussed in Section II, the basic settings for mobile cloud resources include the CPU process frequency of the mobile device/cloud server, the idle power of the mobile device, computation power of the mobile device/cloud server and so on, all described here in Table 1. The environment parameters, such as f_M and f_C, can be obtained by the task manager of the system. Besides, we use Powertutor to monitor the real-time power consumption of the mobile device and calculate the average data as the environment parameters in the simulation, including P_0, P_M, P_{up}, P_{down}. The value of uplink/download rate R_{up}/R_{down} can be monitored by a software kit. In this article, we use NetTraffic to monitor and record the rate by uploading/downloading 100 pictures in real time and calculate its average as the parameters in the simulation.

TABLE 1. Parameters of the Environment Used in simulation.

Parameters	values
CPU frequency of the mobile device(f_M) (GHz)	1.7
CPU frequency of the cloud (f_C) (GHz)	3.6
Idle power of the mobile device (P_0) (W)	0.001
Computation power of the mobile device (P_M) (W)	1.2
Computation power of the cloud (P_C) (W)	30
Data uploading power of the mobile device (P_{up}) (W)	0.5
Data downloading power of the mobile device (P_{down}) (W)	0.1
Uploading rata of the mobile (R_{up}) (kb/s)	6000
Downloading rata of the mobile (R_{down}) (kb/s)	1000

4) PARAMETER SETTINGS FOR ALGORITHMS

In GA, we design 30 individuals in a population during each iteration. The initial chromosome is generated randomly. New individuals will be selected according to their fitness value with a probability of their contribution to the quality of the whole population. We use roulette wheel selection strategy to selection the new individuals. The gene operators: crossover rate is 0.6 and mutation is 0.1. Two-point crossover method is applied to generate offspring. The maximum iteration is 100 times.

In DPSO and EERA, 20 particles are created in each iteration and the initial position is generated randomly. The maximum iteration is 100 times. To guarantee the diversity of the population, the learning parameter for self-experience is set as 0.3 and the learning parameter for global experience is set as 0.5. The best particle position is the position with minimum UtilityCost. All the algorithms are running 10 times with the same task graph and parameters and the average results are regarded as the final results.

C. EXPERIMENTAL RESULTS

In this section, the simulation experimental results of the proposed different heuristic/metaheuristic algorithms on different basic measurement are demonstrated. Since the UtilityCost of the workflow is proposed for balancing the goal of minimizing the total energy consumption and goal of minimizing the total execution time, thus the first key measurement of their performance is the UtilityCost of the workflow (**UC**). Meanwhile, in order to compare and analyze the performance in more detail, we also employ two other key measurements including the Energy Consumption of Mobile Device (**ECMD**), and the ratio of the workflow Execution Time to the Workflow Deadline (**ET/WD**).

1) RESULTS FOR UTILITYCOST OF MOBILE CLOUD WORKFLOW

Fig.5 illustrates the UtilityCost (UC) results of different algorithms on two graph structures. Fig.5(a) and Fig.5(b) show the results on ten different task graphs (from 12 to 102 tasks) when the relaxation factor k is equal to 2. Fig.5(c) and Fig.5(d) show the results on ten different relaxation factors (from 1 to 10) when the workflow scale n is equal to 42. From the definition of UtilityCost, we can see that the UC value of Local is equal to 1 all the time and hence it is represented by a straight line. The results show that the smaller the value of UC, the better the performance of these algorithms is.

From Fig.5(a) and Fig.5(b), we can see that the UC values by all algorithms are lower than 1 for the balanced structure. However, the UC values by GA and DPSO fluctuates significantly, for example, it is close to 0.8 in the scenario with 22 and 32 tasks by GA. Such a phenomenon indicates that the process of finding the optimal solution is unstable in GA and DPSO as both of them are employing random search process. And for the general structure, the UC value is bigger than 1 with the smaller workflow size. For example, in the scenario with 22 tasks, the UC value by GA is about 1.41 and the UC value by DPSO is about 1.10 which are both larger than 1. Except for them, the UC value by other algorithms including HEFT, GE, GU, EERA all equals to 1 which indicates the optimal resource allocation decision in this scenario is Local.

Clearly, when the workflow size is larger, the random search algorithms become more efficient. In general, for the balanced structure, the rank for the efficiency in minimizing UC with different workflow sizes is EERA>GE>GU>HEFT>DPSO>GA; for the general structure, the rank is GE>EERA>GU>DPSO>HEFT>GA.

From Fig.5(c) and Fig.5(d), we can see that with the same workflow size $n = 42$, the UC values by these algorithms are very different. For example, in the scenario with $k = 3$ (balanced structure), the UC value by GA is about 0.96, HEFT is about 0.38, GE is about 0.40, GU is about 0.63, DPSO is about 0.35 and EERA is about 0.58. The difference between the maximum value by GA and the minimum value by DPSO is 0.61. And for the general structure, in the scenario with $k = 8$, the UC value by GA is about 0.62, HEFT is about 0.28, GE is about 0.37, GU is about 0.37, DPSO is about 0.44 and EERA is about 0.08. The difference between the maximum value by GA and the minimum value by EERA is 0.53.

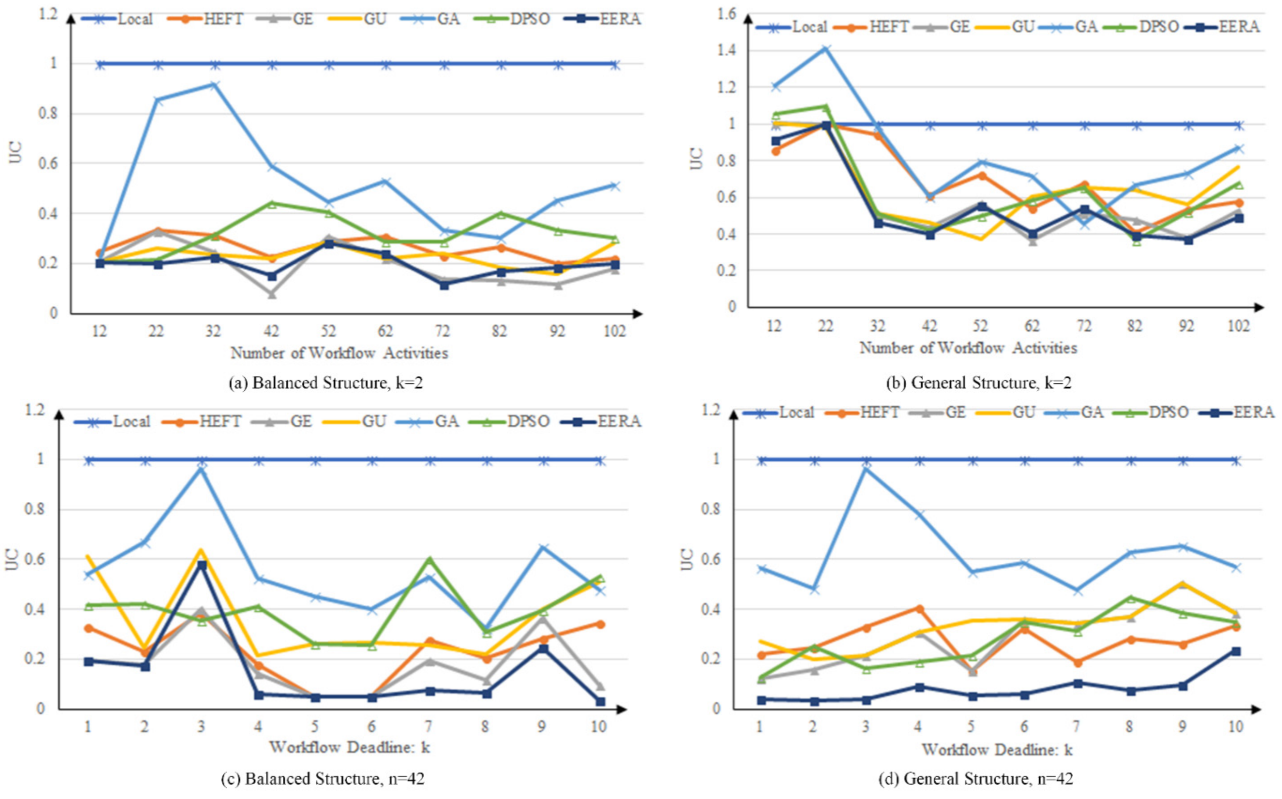


FIGURE 5. Comparison results on UtilityCost (UC).

In general, for the balanced structure, the rank for the efficiency in minimizing UC with different relaxation factors is EERA>GE>HEFT>GU>DPSO>GA; for general structure, the rank is EERA>HEFT>DPSO>GU>GE>GA. Therefore, our EERA algorithm can achieve the best performance among all these algorithms in minimizing the value of UC.

2) RESULTS FOR ENERGY CONSUMPTION OF MOBILE DEVICE

Fig.6 illustrates the energy consumption results of different algorithms on two graph structures. Fig.6(a) and Fig.6(b) show the results on ten different task graphs (from 12 to 102 tasks) when the relaxation factor k is equal to 2. Fig.6(c) and Fig.6(d) show the results on ten different relaxation factors (from 1 to 10) when the workflow scale n is equal to 42 (namely 42 tasks).

From Fig.6(a) and Fig.6(b), the energy consumption of the mobile device is increasing progressively with "Local" when n becomes bigger, which is consistent with the real-world situation. All heuristic/metaheuristic-based algorithms can reduce the energy consumption more effectively on the balanced structure than the general structure. When the workflow size is small, the results of energy consumption by HEFT, GE, GU, DPSO and EERA are similar except for GA. For example, in Fig.6(a) with 22 tasks, the energy consumption by Local is about 1.81J, GA is about 1.48J, HEFT is about 0.86J, GE is about 0.66J, GU is about 0.66J, DPSO is about 0.74J and EERA is about 0.49J. When the workflow size becomes larger, the difference of energy consumption by different algorithms is becoming more apparent. For example, in the ninth scenario with 92 tasks in Fig.6(a), the energy consumption by Local is about 7.51J, GA is about 4.21J, HEFT is about 2.55J, GE is about 1.25J, GU is about 2.46J, DPSO is about 3.05J and EERA is about 2.16J. In general, the performance of GA is the worst among them and the performance of our EERA is excellent but second to GE. For the general structure, the energy consumption by all heuristic/metaheuristic-based algorithms have also decreased compared with Local but not as significantly as for the balanced structure. For example, in the scenario with 102 tasks for general structure, the largest one by Local is about 10.88J, and the smallest one by EERA is about 7.58J, which is a reduction of 30.33%. But for the balanced structure, in the same scenario with 102 tasks, the largest one by Local is about 8.33J, the smallest one by GE is about 1.73J, and our proposed EERA is about 2.58J which is 69.03% lower than the highest one and 31.9% lower than the DPSO algorithm.

In general, all heuristic/metaheuristic-based algorithms can reduce the energy consumption. Specifically, for the balanced structure, the rank for the efficiency in energy saving with different workflow sizes is GE>EERA>GU>HEFT>DPSO>GA; while for the general structure, the rank is GE>EERA>GA>DPSO>HEFT>GU.

From the Fig.6(c) and Fig.6(d), with the same workflow size and different relaxation factor, the performance

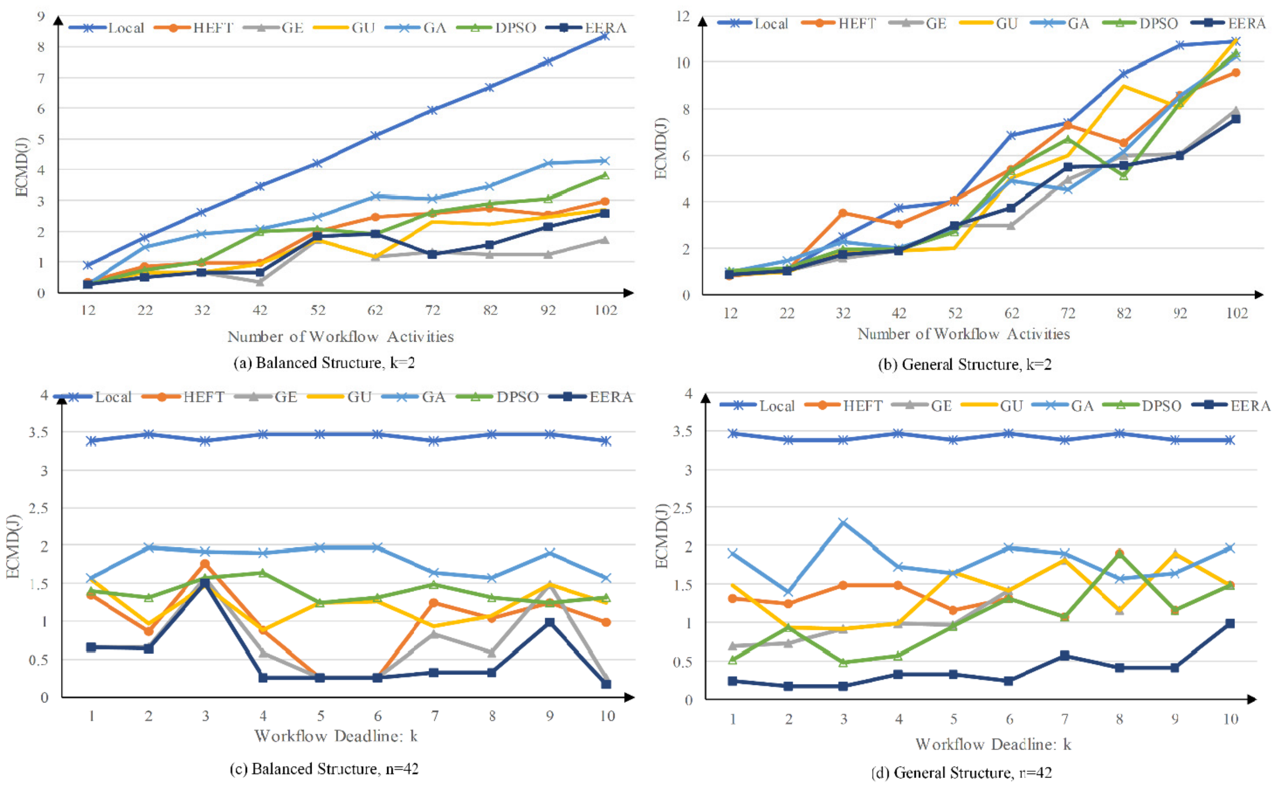


FIGURE 6. Comparison results on energy consumption of mobile device (ECMD).

by different algorithms are relatively stable, which indicates that the relaxation factor k has a limited influence on the energy consumption of the mobile device. Specifically, for balanced structure, the rank for the efficiency in energy saving with different relaxation factors is EERA>GE>HEFT>GU>DPSO>GA, while for the general structure, the rank is EERA>DPSO>GE>GU>HEFT>GA.

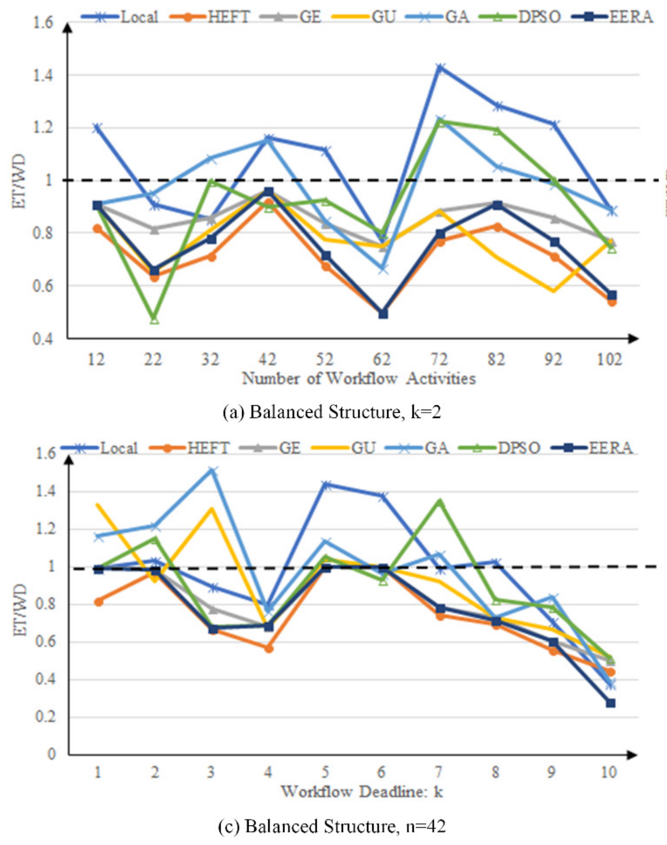
To sum up, the resource allocation strategy based on our proposed EERA algorithm achieves the best overall performance in reducing the energy consumption of the mobile device.

3) RESULTS FOR EXECUTION TIME OF MOBILE CLOUD WORKFLOW

Fig.7 illustrates the execution time results of different algorithms on two graph structures. To focus on the comparison results, we present the results of ET/WD (namely the value of the Execution Time divided by the Workflow Deadline) instead of the execution time itself. Fig.7(a) and Fig.7(b) show the results on ten different task graphs (from 12 to 102 tasks) when the relaxation factor k is equal to 2. Fig.7(c) and Fig.7(d) show the results on ten different relaxation factors (from 1 to 10) when the workflow scale n is equal to 42. The dashed line represents the case where the execution time is equal to the workflow deadline, namely the value is equal to 1.

From Fig.7(a) and Fig.7(b), we can see the execution time by Local, GA, DPSO could be more easily exceeding the workflow deadline, while the other algorithms can meet the deadline. Specifically, for the balanced structure, in 6 scenarios with 12, 42, 52, 72, 82 and 92 tasks, the workflow cannot meet the deadline constrains by Local. In 4 scenarios with 32, 42, 72 and 82 tasks, the workflow cannot meet the deadline constrains by GA. In 2 scenarios with 72 and 82 tasks, the workflow cannot meet the deadline constrains by DPSO. In all other scenarios, the workflow can be delivered within the deadline by algorithms including HEFT, GE, GU and EERA. As for the general structure, GA and Local have 8 scenarios where the deadline constraints cannot be met ($n>22$), while the DPSO only fails in 1 scenario ($n = 72$). Except for these scenarios, all deadline constraints can be met by algorithms.

We reckon the reason for the above results is because the optimization target of both GA and DPSO is the UtilityCost rather than the deadline. The optimization target of HEFT is minimizing the execution time of all workflow tasks which implicitly guarantees the satisfaction of deadline constraints. Among them, our EERA algorithm achieves the best performance because it can guarantee the satisfaction of deadline constraints through the calculation of PCPs. In general, for the balanced structure, the rank for the efficiency in minimizing the workflow execution time with different workflow sizes



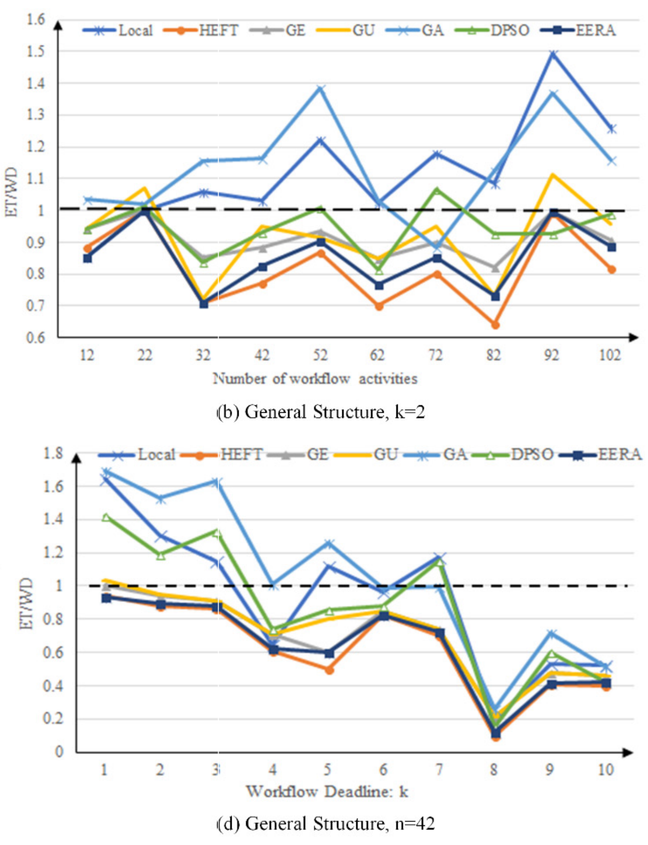


FIGURE 7. Comparison results on the ratio of execution time to workflow deadline (ET/WD).

is GE>EERA>GU>HEFT>DPSO>GA; for the general structure, the rank the rank is GE>EERA>GA>DPSO> GU>GA.

From Fig.7(c) and Fig.7(d), we can see that with the same workflow size $n = 42$, the ratio of deadline violation is higher when the relaxation factor becomes smaller. When the relaxation factor becomes larger, the performance of all algorithms is better. For the balanced structure, when $k>7$, all algorithms can meet the deadline constraints. For example, in the scenario with $k = 8$, the ET/WD by Local is about 1.02, GA is about 0.73, HEFT is about 0.69, GE is about 0.73, GU is about 0.73, DPSO is about 0.83 and EERA is about 0.71. For the general structure, the ratio of ET/WD decreased fast and its average is lower than the balanced structure. When $k>3$, except for 2 scenarios ($k = 5$ by GA, $k = 7$ by DPSO), the total execution time of the workflow can meet the deadline in all the other cases. For example, in the scenario with $k = 8$, the ET/WD by Local is about 0.19, GA is about 0.26, HEFT is about 0.41, GE is about 0.47, GU is about 0.47, DPSO is about 0.15 and EERA is about 0.12.

In general, for the balanced structure, the rank for the efficiency in minimizing the workflow execution time with different relaxation factors is HEFT>EERA>GE>DPSO> GU>GA; for the general structure, the rank is HEFT> EERA>GE>GU>DPSO>GA.

4) COMPARISON RESULTS BETWEEN EERA AND DPSO

The proposed EERA algorithm is established based on the PCPs and DPSO algorithm, thus in this part, we compare EERA algorithm with DPSO algorithm individually to demonstrate the performance of EERA algorithm. The average comparison results between proposed EERA algorithm and DPSO are illustrated in Table 2, including three evaluation measurements: UC, ECMD and ET/WD.

From Tab.2, we can see that when the relaxation factor k is fixed to 2, the average UC, ECMD and ET/WD value of EERA algorithm from a task graph with 12 nodes to a task graph with 102 nodes are all improved on two different graph structures. At the same time, when the task graph size is fixed to 42 nodes, the average UC, ECMD and ET/WD value of EERA algorithm with different relaxation factor k from 1 to 10 are also improved on two different graph structures. The results have proved that the proposed EERA algorithm performs better than the DPSO algorithm in improving the UtilityCost of the MCC system, reducing the energy consumption of the mobile device and speeding up the response time of the workflow.

5) SUMMARY OF EXPERIMENTAL RESULTS

Given the experimental results illustrated above, it is obvious that there is not a single algorithm which has the

TABLE 2. The average comparison results (EERA vs DPSO).

	k=2		n=42	
	balanced	general	balanced	general
UC	34.67%↑	11.99%↑	59.03%↑	70.00%↑
ECMD	38.52%↑	11.83%↑	61.63%↑	61.23%↑
ET/WD	41.43%↑	12.20%↑	13.67%↑	23.40%↑

best performance in every situation. Many factors, such as workflow size, the target of optimization, and the workflow structure can affect the performance of these resource allocation algorithms. To sum up, we can have the following conclusions:

- The workflow size has the most impact on the performance of these algorithms, the graph structure is second, and the relaxation factor has the least impact.
- When the workflow size is larger, the performance of GA and DPSO becomes much dynamic. In contrast, our EERA can maintain stable performance and in most cases is the best one in both reducing the energy consumption and meeting the workflow deadline. Accordingly, EERA can achieve the best performance under the measurement of UC.

Therefore, for mobile cloud workflow, we recommend the green resource allocation strategy based on our proposed EERA algorithm which can achieve the best performance among all representative algorithms discussed in this article.

VI. RELATED WORK

In mobile cloud computing, response time is not the only target for mobile applications, reducing the energy consumption of the mobile device is also one of the most important QoS dimensions which has attracted a lot of efforts from the researchers [21], [22].

Earlier research works focus on designing computation offloading frameworks and strategies to dynamically offload computing tasks from the resource-constrained mobile devices to the nearby cloud computing servers to improve the service quality or reduce the energy consumption of the mobile device. CloneCloud [23], ThinkAir [24], Cloudlet [9] and Jade [25] are pioneered and fundamental frameworks which use computation offload as a main approach to improve the QoS of mobile applications. These frameworks can partition the whole application into several sub-tasks in coarse or fine granularity so as to achieve different task offloading objectives such as decreasing execution cost, maximizing throughput, reducing network latency or minimizing energy consumption. Additionally, many task offloading approaches have been proposed to allocate the cloud resources and the mobile resources to the mobile applications based on task partition [26], [27]. However, most of these studies focus on independent mobile applications are not suitable for mobile cloud workflow applications introduced in this article. Mobile cloud workflow is a mobile application that can be recognized as a set of tasks in a partial order to achieve a specific business goal. As discussed in the Introduction section, mobile cloud workflow applications often need human intervention to complete specific tasks. In addition, a mobile cloud workflow system may involve a group of devices with access to the cloud resources which act as not only a service provider, but as a main computation offloading handler. These features bring many challenges for the resource management in mobile cloud workflow systems.

Recently, a lot of attention has been paid to design energy efficient resource allocation strategies for mobile cloud workflow based on those frameworks mentioned above [27], [28]. These strategies can be categorized into three types based on their objectives: enhancing QoS, saving energy or the hybrid [6], [10], [16], [29]–[31]. The work in [37] considers the allocation problem in an edge cloud computing system as an M/M/c queue and solves the problem form two angles: the flat deployment and the hierarchical deployment to minimize the overall average response time of the system applications, which doesn't consider the system energy consumption. The work in [10] systematically analyzes the characteristics of cloud-assisted mobile application workflows and indicates that the task allocation problem becomes a critical step in deciding the energy-footprint of the workflow. The authors construct a quadratic binary program to model the task allocation problem in MCC and then propose an implementation of the simulated annealing algorithm and the greedy autonomous offload algorithm to find the approximate optimal solution with minimum energy, which ignores the execution time of the workflow. The work in [16] investigates the offloading problem which takes both the energy consumption of the mobile device and the execution delay of the application into consideration by formulating the problem into a 0-1 integer linear programming (ILP) problem. The authors in [31] model the task scheduling problem in MCC as an energy consumption optimization problem and propose three different heuristic algorithms, such as greed-based, group-based and genetic-based task scheduling algorithms, to solve it. Guo S. *et al.* models the offloading problem as an energy-efficiency cost (EEC) minimization problem with task-dependency and time deadline constraints. And proposes a dynamic and distributed eDors algorithm with three steps: offloading selection, clock frequency control and power allocation [6], [32]. These works above all take the energy consumption of the end device as their optimization goals but they ignore the task deadlines and the location constraints introduced above in the paper. Yadav R. *et al.* proposed an adaptive heuristics algorithm, specifically creates an upper CPU utilization threshold to detect overloaded hosts and dynamic VM selection algorithms to consolidate the VMs, to minimize the total energy consumption and maximize Quality of Service in mobile cloud computing [35]. However, the work has focused on the single task but do not considered the workflow tasks. And furthermore, the mobile device, plays only a client role in the paper, has not participated in the execution of the application task. The work in [36] proposes a novel Delay-constraint and Reverse Auction-based Incentive Mechanism, named DRAIM to maximize the revenue of Mobile Network Operator. Although the mobile device has participated in the task execution, its optimization goal is maximizing the revenue of Mobile Network Operator which is different with this article. For scientific workflow in MCC, the work in [29] propose a two-phases algorithm with a cost adaptive VM management under the constraints of deadline and budget, which focuses on the Service Level Agreement contains the deadline and the resource usage and is not consider the energy consumption. A mobility-enabled and fault-tolerance offloading system is proposed in [30] to make the offloading strategies for mobile service workflow. These approaches have considered the location constraints of the specific tasks but do not consider the task deadline constraints result from the uncertainties during the execution of the workflow.

The energy-efficient resource allocation strategy for mobile cloud workflows in this article takes both the task deadlines and the location constraints into consideration. We systematically analyze and formulate the allocation problem into a minimization problem with the goal of minimizing the UtilityCost to balance the conflict between the energy consumption of the mobile device and the execution time of the whole workflow.

VII. CONCLUSION AND FUTURE WORK

In this article, to achieve an optimal balance between the energy consumption and the QoS of the mobile cloud workflow application, we introduced a novel concept of "UtilityCost" to represent the trade-off between energy consumption and execution time, and formulated the resource allocation problem into an optimization model with the goal of minimizing the UtilityCost while meeting the execution time constraints for mobile cloud workflow. Further, we proposed a new energy-efficient resource allocation algorithm named EERA to find the approximate optimal offloading decision. Comprehensive simulation experiments have demonstrated that our proposed algorithm outperformed all other representative algorithms including Greedy (GU and GE), HEFT, GA and DPSO under various parameter configurations and different workflow graph structures.

As dynamic network bandwidth can have a significant impact on the data communication time, in the future, we will investigate the effect of dynamic bandwidth between the mobile device and the cloud on resource allocation decisions and improve our strategy to be more adaptive and robust to network related issues.

REFERENCES

[1] Cisco, "Visual networking index: Global mobile data traffic forecast update 2017–2022," Cisco, San Jose, CA, USA, White Paper white-paper-c11-738429, 2019. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[2] X. Qiao, P. Ren, G. Nan, L. Liu, S. Dustdar, and J. Chen, "Mobile Web augmented reality in 5G and beyond: Challenges, opportunities, and future directions," *China Commun.*, vol. 16, no. 9, pp. 141–154, Sep. 2019.

[3] P. Riti, "Mobile development," in *Practical Scala DSLs*. Berkeley, CA, USA: Apress, 2018, pp. 139–158.

[4] A.-L. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 45–57, Mar. 2018.

[5] D. C. Marinescu, "Big data, data streaming, and the mobile cloud," in *Cloud Computing*. San Mateo, CA, USA: Morgan Kaufmann, 2018, ch. 12, pp. 439–487.

[6] S. Guo, J. Liu, Y. Yang, B. Xiao, and Z. Li, "Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 18, no. 2, pp. 319–333, Feb. 2019.

[7] H. Debnath, G. Gezzi, A. Corradi, N. Gehani, X. Ding, R. Curtmola, and C. Borcea, "Collaborative offloading for distributed mobile-cloud apps," in *Proc. 6th IEEE Int. Conf. Mobile Cloud Comput., Services, Eng. (MobileCloud)*, Bamberg, Germany, Mar. 2018, pp. 87–94.

[8] H. Yeganeh, A. Salahi, and M. A. Pourmina, "A novel cost optimization method for mobile cloud computing by capacity planning of green data center with dynamic pricing," *Can. J. Electr. Comput. Eng.*, vol. 42, no. 1, pp. 41–51, Apr. 2019.

[9] M. Whaiduzzaman, A. Naveed, and A. Gani, "MobiCoRE: Mobile device based cloudlet resource enhancement for optimal task response," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 144–154, Jan. 2018.

[10] B. Gao, L. He, X. Lu, C. Chang, K. Li, and K. Li, "Developing energy-aware task allocation schemes in cloud-assisted mobile workflows," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, Liverpool, U.K., Oct. 2015, pp. 1266–1273.

[11] N. K. Shukla, R. Pila, and S. Rawat, "Utilization-based power consumption profiling in smartphones," in *Proc. 2nd Int. Conf. Contemp. Comput. Informat. (ICI)*, Noida, India, Dec. 2016, pp. 881–886.

[12] S. Jukna and H. Seiwert, "Greedy can beat pure dynamic programming," *Inf. Process. Lett.*, vol. 142, pp. 90–95, Feb. 2019.

[13] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.

[14] D. Kumar, B. Sahoo, B. Mondal, and T. Mandal, "A genetic algorithmic approach for energy efficient task consolidation in cloud computing," *Int. J. Comput. Appl.*, vol. 118, no. 2, pp. 1–6, May 2015.

[15] S. J. Nirmala and S. M. S. Bhanu, "Catfish-PSO based scheduling of scientific workflows in IaaS cloud," *Computing*, vol. 98, no. 11, pp. 1091–1109, Jun. 2016.

[16] X. Wang, J. Wang, X. Wang, and X. Chen, "Energy and delay tradeoff for application offloading in mobile cloud computing," *IEEE Syst. J.*, vol. 11, no. 2, pp. 858–867, Jun. 2017.

[17] P. Balakrishnan and C. K. Tham, "Energy-efficient mapping and scheduling of task interaction graphs for code offloading in mobile cloud computing," in *Proc. IEEE/ACM Int. Conf. Utility Cloud Comput.*, Dec. 2013, pp. 34–41.

[18] W. Zhang and Y. Wen, "Energy-efficient task execution for application as a general topology in mobile cloud computing," *IEEE Trans. Cloud Comput.*, vol. 6, no. 3, pp. 708–719, Jul. 2015.

[19] P. Zhao, H. Tian, and B. Fan, "Partial critical path based greedy offloading in small cell cloud," in *Proc. IEEE VTC*, Sep. 2016, pp. 1–5.

[20] Z. Zhang, J. Wu, G. Jiang, L. Chen, and S. Lam, "QoE-Aware task offloading for time constraint mobile applications," in *Proc. IEEE 42nd Conf. Local Comput. Netw. (LCN)*, Singapore, Oct. 2017, pp. 510–513.

[21] B. Zhou and R. Buyya, "Augmentation techniques for mobile cloud computing: A taxonomy, survey, and future directions," *ACM Comput. Surveys*, vol. 51, no. 1, pp. 1–38, Apr. 2018.

[22] S. Deng, L. Huang, H. Wu, and Z. Wu, "Constraints-driven service composition in mobile cloud computing," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, San Francisco, CA, USA, Jun. 2016, pp. 228–235.

[23] B. G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst. (EuroSys)*, New York, NY, USA, 2011, pp. 301–314.

[24] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[25] H. Qian and D. Andresen, "Extending mobile Device's battery life by offloading computation to cloud," in *Proc. 2nd ACM Int. Conf. Mobile Softw. Eng. Syst.*, Florence, Italy, May 2015, pp. 150–151.

[26] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[27] H. Kanemitsu, M. Hanada, and H. Nakazato, "Multiple workflow scheduling with offloading tasks to edge cloud," in *Proc. Int. Conf. Cloud Comput. (Cloud)*, San Diego, CA, USA, 2019, pp. 38–52.

[28] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[29] W.-J. Kim, D.-K. Kang, S.-H. Kim, and C.-H. Youn, "Cost adaptive VM management for scientific workflow application in mobile cloud," *Mobile Netw. Appl.*, vol. 20, no. 3, pp. 328–336, Jun. 2015.

[30] S. Deng, L. Huang, J. Taheri, and A. Y. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3317–3329, Dec. 2015.

[31] C. Tang, M. Hao, X. Wei, and W. Chen, "Energy-aware task scheduling in mobile cloud computing," *Distrib. Parallel Databases*, vol. 36, no. 3, pp. 1–25,2018.

[32] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, Apr. 2016, pp. 1–9.

[33] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," *J. Parallel Distrib. Comput.*, vol. 70, no. 1, pp. 13–22, Jan. 2010.

[34] F. Sun, F. Hou, N. Cheng, M. Wang, H. Zhou, L. Gui, and X. Shen, "Cooperative task scheduling for computation offloading in vehicular cloud," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 11049–11061, Nov. 2018.

[35] R. Yadav and W. Zhang, "MeReg: Managing energy-SLA tradeoff for green mobile cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2017, no. 2, pp. 1–11, 2017.

[36] H. Zhou, X. Chen, S. He, J. Chen, and J. Wu, "DRAIM: A novel delay-constraint and reverse auction-based incentive mechanism for WiFi offloading," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 4, pp. 711–722, Apr. 2020.

[37] E. Wang, D. Li, B. Dong, H. Zhou, and M. Zhu, "Flat and hierarchical system deployment for edge computing systems," *Future Gener. Comput. Syst.*, vol. 105, pp. 308–317, Apr. 2019.

JUAN LI received the M.S. and Ph.D. degrees from the Computing Science School, Wuhan University, Wuhan, China, in 2014 and 2018, respectively. Her research interests include mobile edge computing, mobile cloud computing, workflow scheduling, and resource allocation technology.

XIAOLU XU received the M.S. degree from the School of Power and Mechanical Engineering, Wuhan University, Wuhan, China, in 2013. His research interests include resource allocation algorithm research in electrical power systems, and the application design of mobile cloud application in smart grid.