IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 21, 2021, accepted February 22, 2021, date of publication March 2, 2021, date of current version March 10, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3063456

Task Duplication-Based Scheduling Algorithm for Budget-Constrained Workflows in Cloud Computing

FUGUANG YAO, CHANGJIU PU, AND ZONGYIN ZHANG
Information Center, Chongqing University of Education, Chongqing 400065, China

Corresponding author: Fuguang Yao (yaofuguang@cque.edu.cn)

This work was supported in part by the Scientific Research Platform of Chongqing University of Education through the Big Data Practice Platform for Sports and Health Based on Campus Internet of Things (IOT) under Grant 2017XJPT07, in part by the Future School (Infant Education) of the National Center for Schooling Development Program of China through the Research on the Construction of Intelligent Kindergarten Based on IOT under Grant CSDP18FC3204, in part by the Chongqing Key Research Base of Humanities and Social Sciences (Chongqing Research Center of Overall Development of Urban-Rural Teachers Education) through the Research on Post-Service Training Management and Program for Primary and Secondary School Teachers in Chongqing Based on Big Data under Grant 18JDZDWT04, in part by the Project the Key Issue of Chongqing Education Science (13th Five-Year Plan in 2018) through the Research and Practice of Intelligent Classroom Construction Based on Internet of Things and Big Data under Grant 2018-GX-017, and in part by the Scientific Research Program of Chongqing University of Education, Research and Design of Intelligent Wristbands Application System under Grant KY2016TZ02.

ABSTRACT Workflow scheduling is crucial to the efficient operation of cloud platforms, and has attracted a lot of attention. Up to now, many algorithms have been reported to schedule workflows with budget constraints, so as to optimize workflows' makespan on cloud resources. Nevertheless, the hourly-based billing model in cloud computing is an ongoing challenge for workflow scheduling that easily results in higher makespan or even infeasible solutions. Besides, due to data constraints among workflow tasks, there must be a lot of idle slots in cloud resources. Few works adequately exploit these idle slots to duplicate tasks' predecessors to shorten their completion time, thereby minimizing workflow's makespan while ensuring its budget constraint. Motivated by these, we propose a task duplication based scheduling algorithm, namely TDSA, to optimize makespan for budget-constrained workflows in cloud platforms. In TDSA, two novel mechanisms are devised: 1) a dynamic sub-budget allocation mechanism, it is responsible for recovering unused budget of scheduled workflow tasks and redistributing remaining budget, which is conducive to using more expensive/powerful cloud resources to accelerate completion time of unscheduled tasks; and 2) a duplication-based task scheduling mechanism, which strives to exploit idle slots on resources to selectively duplicate tasks' predecessors, such advancing these tasks' completion time while trying to ensuring their sub-budget constraints. At last, we carry out four groups of experiments, three groups on randomly generated workflows and another one on actual workflows, to compare the proposed TDSA with four baseline algorithms. Experimental results confirm that the TDSA has an overwhelming superiority in advancing the workflows' makespan (up to 17.4%) and improving the utilization of cloud computing resources (up to 31.6%).

INDEX TERMS
Cloud computing, task duplication, workflow scheduling, resource provision, heuristic mechanism.

I. INTRODUCTION

As a new computing paradigm, cloud computing provides end-users with highly scalable applications, platforms, and hardware as services through the Internet [1]. In this paradigm, end-users can access resources on-demand using the "pay-as-you-go" mode, and just pay for their actual resource usage, thus significantly reducing the cost and operation expenses [2]. Relying on these advantages, cloud computing developed rapidly in recent years, and has been widely used for processing big data applications coming from various fields, such as astronomy [3], [4], healthcare [5],

The associate editor coordinating the review of this manuscript and approving it for publication was P.K. Gupta.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

bioinformatics [6], intelligent transportation [7], and Internet of Things [8], [9].

In many fields, running big data applications often has the following challenging characteristics. An application involves a series of interdependent stages, and each stage contains a large number of independent tasks, where there exist data interactions among tasks belonging to different stages [10], [11]. For instance, an image processing application can be divided into many dependent stages, such as preprocess, segment, and object classification [12], [13]. Each stage contains a large number of tasks, and the input data of these tasks is the output data of the tasks in the previous stage [13]. These applications are termed as workflows in distributed community [14], and one workflow can be formulated as a Directed Acyclic Graph (DAG), where nodes stand for its tasks and edges indicate the data dependencies among tasks [15].

Workflow scheduling is of importance to achieve high performance for heterogeneous cloud platforms [16], and it includes two inseparable segments, i.e., resource provisioning and task scheduling [17]. The former refers to the decision of resource enrolling, including type, quantity, and usage time [18]. The latter is to map all tasks to the enrolled resources and sort tasks on each resource. It has been deeply recognized that workflow scheduling is essential for efficient operation of cloud computing platforms, such as shortening makespan of workflows, ensuring users' budget constraints, and improving resource utilization. Generally, scheduling workflows in cloud is NP-complete [2], [19]. A workflow contains hundreds or even more tasks, and the feasible solution space on elastic cloud platforms increases exponentially with the number of tasks, so searching the optimal schedule is often prohibitively expensive. Therefore, designing heuristic algorithms with low-complexities has become an attractive choice to generate quick and efficient schedules within reasonable time [11], [20].

Budget constraint has become one of the major concerns for implementing workflows on cloud computing [17], [21], [22]. In recent years, a series of works on scheduling budget-constrained workflows in clouds have been published [16], [20], [23]–[28]. For instance, Arabnejad *et al.* designed a deadline-budget constrained workflow scheduling algorithm with low time-complexity to search feasible schedules accomplishing both constraints of deadline and budget [29]. Faragardi *et al.* developed a greedy resource provisioning mechanism to extend the classical Heterogeneous Earliest-Finish-Time (HEFT) algorithm [30] to minimize workflows' makespan while ensuring their budget constraints [14]. Sun *et al.* considered both the budget and deadline constraints of workflows, and defined a new sub-deadline for each task to improve the HEFT method for workflow scheduling [31]. Rizvi *et al.* designed a fair scheduling algorithm for shortening workflows' makespan while ensuring their budget constraints [32]. Zuo *et al.* developed a time- and cost-first mechanism to improved the ant colony optimization based scheduling approach for optimizing task finish time and cost when considering their deadline and cost constraints [33]. Zhou *et al.* defined a balance factor for workflow tasks based on their optimistic spare deadlines and budgets, and designed a resource selection approach for workflow tasks to improve the possibility of ensuring workflows' deadline and budget requirements [34]. Nevertheless, these existing works ignore the time slots in resources left by data dependencies among workflow tasks, and do not exploit the advantages of task duplication to enhance tasks' start and finish time. Besides, due to the heterogeneous and diverse workflow structure, how to distribute the overall budget of a workflow to all its tasks is an ongoing challenge.

Until now, some task duplication/replication-based scheduling approaches have been proposed for scheduling workflows in cloud computing [25], [35]–[37]. For instance, Casas *et al.* designed a file reuse-replication scheduling approach to divide a workflow into many sub-workflows and employ file replication mechanisms to reduce the number of files transferred among tasks, such making a trade-off between makespan time and cost [38]. To ensure reliability constraints or improve the fault-tolerant capacity, task replication mechanisms were designed for scheduling workflows in cloud computing [39]–[41]. But, these two works did not consider the budget constraint of workflows, and the task replication mechanisms tend to replicate most tasks multiple times, regardless of whether existing slots are available, which increases resource usage significantly and makes it easier to cost more than workflows' budget. Besides, to relieve the waste of time slots in resources, Chen *et al.* designed a task duplication based scheduling approach to alleviate the time overheads of data encryption for a single workflow having security or deadline requirements but without any budget constraints [2], [10].

Due to the data constraints between tasks, even with efficient workflow scheduling algorithms, there will inevitably be a lot of time slots on heterogeneous cloud resources. It is important to note that if a task runs on the same resource as its precursor, the task can use this precursor's output data directly, without data transfer. Thus, duplicating the precursors of a task to existing time slots on the same resource can avoid the data transfer delay to advance this task's start and finish time without additional resource usage. So it motivates us to explore how to duplicate the precursors when scheduling a task, so as to advance its start/finish time and improve resource utilization.

Besides, users typically specify the budget for the entire workflow, not for individual tasks. However, workflow scheduling needs to be done task by task. So when scheduling a task, how much budget can be used to improve its start and completion time without causing all the tasks to cost more than the budget is also a challenge. So it motivates us to explore how to reasonably allocate a sub-budget for each task.

Keep the aforementioned two motivations in mind, to minimize makespan for workflows under budget limits, we design a task duplication based workflow scheduling algorithm with the following two new contributions.

(1) We design a budget allocation mechanism to dynamically distribute sub-budgets for all the un-scheduled workflow tasks. This mechanism is able to not only avoid single task costing too much to ensure workflow's overall budget constraint, but also recover unused sub-budget of scheduled tasks to use more expensive/powerful cloud resources to accelerate completion of unscheduled tasks.

(2) A duplication-based task scheduling mechanism is designed to asymptotically duplicate a task's precursors to the existing time slots on the same resource, such advancing its completion time while trying to ensuring its sub-budget constraint. The effectiveness of the proposed algorithm is verified by comparing it with the four baseline algorithms on the context of various scenarios.

The rest of this article is organized as follows. Section II describes the optimization problem of scheduling workflows in cloud computing. Then, the proposed scheduling algorithm is detailed in Section III, followed by the experimental verification of the algorithm in Section IV. Next, Section IV concludes this article and presents two points worthy of further research.

II. PROBLEM DESCRIPTION

In this section, we introduce the models for elastic heterogeneous cloud resources, workflow, and optimization problems.

A. CLOUD PLATFORM

Similar to [14], [42], [43], this article also pays close attention to the Infrastructure-as-a-Service (IaaS) mode. In this mode, various types of resources are provided, and each type of resource share the same configurations, such as price, CPU, memory size, bandwidth, etc. Suppose that the cloud platform provides m types of resources, the set $U = \{1, 2, \cdots, m\}$ represents all resource types, in which $u \in U$ indicates the u-th type. For a resource type, its price is expressed as $Pr(u)$. In cloud computing, the cost of using resources is calculated on an hourly-based cost model. For instance, the price of the first resource type is \$0.35 per hour. If one resource with the first type is used for 61 minutes, the cost is \$0.70.

Thanks to the elasticity of cloud resources, the number of each type of resources can be increased and decreased according to the actual workload. Then, we employ the symbol r_k^u to indicate the k-th resource, whose type is u.

B. WORKFLOW

Workflow is an application composed of a set of data-dependent tasks, and can be indicated as a directed acyclic graph $DAG = \{T, E, B\}$, in which T and E respectively denote the set of tasks and edges among tasks, and B denotes the overall budget. Furthermore, T can be detailed as $T = \{t_1, t_2, \cdots, t_n\}$, where n refers to the number of tasks. For a task $t_i \in T$, the set of its direct precursors and successors are denoted by $P(t_i)$ and $S(t_i)$, respectively.

For an edge $e_{i,j} \in E$, it means that there exists data dependence from task t_i to task t_j, that is to say, only when task t_i has completed and its output data has reached the resource of running task t_j, task t_j can start running. Then, the weight $w(e_{i,j})$ of the edge $e_{i,j}$ represents the amount of data being transferred from t_i and t_j.

C. OPTIMIZATION MODEL

Since different types of resources have different processing power, one task often exhibits different execution time on different types of resources. The symbol $e_{i,k}$ is used to represent the execution time of task t_i on resource r_k^u.

To describe the heterogeneity of connection links among cloud resources, we employ a matrix M with size $|R| \times |R|$ to model the bandwidth between all pairs of resources, where $|R|$ denotes the number of enrolled resources. The element $m_{i,j}$ indicates the bandwidth between resources r_i and r_j. Suppose two tasks t_i and t_j run on distinct resources and the amount of data between them is $w(e_{i,j})$, the data transfer time $dt_{i,j}$ between them can be calculated as:

$$dt_{i,j} = \triangle + \frac{w(e_{i,j})}{m_{r(t_i),r(t_j)}}, \tag{1}$$

where $\triangle$ refers to the transmission startup time, $r(t_i)$ denotes the index of resource running task t_i.

When tasks t_i and t_j run on the same resources, the data transfer time is assumed to be negligible. Thus, when $r(t_i) = r(t_j)$, $dt_{i,j} = 0$ [44].

Due to the data dependencies among workflow tasks, a task t_i can start running after all its direct precursors have been completed and output data of these precursors have been received. Then, it leads to the following constraint:

$$st_{i,r(t_i)} \geq \max_{t_p \in P(t_i)} \{ct_{p,r(t_p)} + dt_{p,i}\}, \tag{2}$$

where $st_{i,r(t_i)}$ and $ct_{p,r(t_p)}$ respectively denotes the start and completion time of tasks on corresponding resources.

To ensure the budget constraint for running workflow, it comes the following constraint:

$$\sum_{k=1}^{|R|} Pr(r_k^u) \cdot wp_k \leq B, \tag{3}$$

where R represents the set of resources enrolled to execute the workflow, and wp_k denotes the enrollment period of the k-th resource.

On the basis of the data dependencies and budget constraints in (2) and (3), this attempts to minimize workflows' makespan. For a workflow, its makespan refers to the maximum completion time of all tasks. Then, we get the optimization objective as follows:

$$\text{Min} \max_{t_i \in T}\{ct_{i,r(t_i)}\}. \tag{4}$$

III. ALGORITHM DESIGN

How to minimize workflows' makespan while ensuring their budget constraints is an ongoing challenge. In this section, a task duplication based workflow scheduling algorithm, namely TDSA, is designed, as illustrated in Algorithm 1. The TDSA strives to fully exploit existing time slots and dynamically re-allocate unused budget to advance the start/completion time of workflow tasks, so as to optimize the makespan of workflows while satisfying their budgets.

Algorithm 1 Overall Process of *TDSA*

```
Input: A workflow $G = (T, E, B)$;  a set of available resource types $U$;
Output: A schedule for the workflow;
1  $R \leftarrow \emptyset$;
2  $T_s \leftarrow \emptyset$;
3  $T_u \leftarrow T$;
4  $T_r \leftarrow$ Select the tasks without any precursors;
5  $remB \leftarrow B$;
6  $usedC \leftarrow 0$;
7  Trigger $DistrSubBudget()$ to distribute subbudgets for all the tasks in $T_u$;
8  while $T_u$ is not empty do
9      $T_n \leftarrow \emptyset$;
10     for $t_i \in T_r$ do
11         $[R, mC] \leftarrow$ Schedule $t_i$ by $TaskSchedule()$;
12         $usedC \leftarrow usedC + mC$;
13         for $t_s \in Suc(t_i)$ do
14             if All the predecessors of $t_s$ have been scheduled then
15                 $T_n \leftarrow T_n \bigcup \{t_s\}$;
16     $T_s \leftarrow T_s \bigcup T_r$;
17     $T_u \leftarrow T_u \setminus T_r$;
18     $T_r \leftarrow T_n$;
19     $remB \leftarrow B - usedC$;
20     Trigger $DistrSubBudget()$ to distribute subbudgets for all the tasks in $T_u$;
```

As shown in Algorithm 1, the inputs of TDSA are a workflow and the set of resource types. After optimization, its output is the schedule for this workflow, mainly including a set of resources used for executing this workflow; mapping between workflow tasks and resources; and task sequencing on each resource.

Before starting, the set of enrolled resources is first initialized as an empty set (line 1). To distinguish the status of tasks during the scheduling process, symbols T_s and T_u are respectively used to represent the set of scheduled and unscheduled tasks (lines 2-3). The T_r represents the set of tasks being ready for scheduling, and all the tasks having not predecessor are selected to initialize it (line 4). After initializing the remaining budget *remB* and used cost *usedC* so far, the function *DistrSubBudget*() will be triggered to allocate sub-budget for each unscheduled tasks. Then, the TDSA goes into the optimization loop, constantly updating the sub-budgets for unscheduled tasks (line 11) and mapping tasks to resources (line 20).

The optimization loop will repeat the following four steps until all tasks have been scheduled. (1) An empty set T_n is initialized to record tasks that become ready during this iteration (line 9). When a task has no precursors or all of its precursors have been mapped to resources, it is called a ready task in this article. (2) The ready tasks in T_r are scheduled by function *TaskSchedule*(), detailed in Algorithm 3, in an one by one way (lines 10-12). Once a task is scheduled, the set of enrolled resources and the marginal cost of completing the task will be returned (line 11). Besides, some successor tasks of the scheduled task may become ready, and they will be selected to update the set T_n (lines 13-15). (3) After all the tasks in T_r are scheduled, the scheduled task set, unscheduled task set, the ready task set, and the remaining budget will be updated (lines 16-19). (4) The sub-budgets for all the unscheduled tasks will be re-allocated by function *DistrSubBudget*(), as detailed in Algorithm 3.

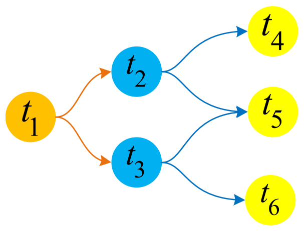

FIGURE 1. **A workflow instance.**

To improve the readability of Algorithm 3, a workflow instance is given in Figure 1 to illustrate some key concepts and operations. Since task t_1 has not precursor, it is a ready task before scheduling. When the operation in line 5 is performed, the set T_r is updated to $T_r = \{t_1\}$. After task t_1 is scheduled, its two successor tasks t_2 and t_3 become ready. Then, by performing the operations from line 13 to 15, the set T_n will record these two ready tasks as $T_n = \{t_2, t_3\}$. Next, by performing the operations from line 16 to 18, the set of scheduled tasks T_s will be updated as $T_s = \{t_1\}$, the set of un-scheduled tasks T_u will be updated as $T_u = \{t_2, t_3, t_4, t_5, t_6\}$, and the set of ready tasks T_r is updated as $T_r = \{t_2, t_3\}$.

Due to complexity in workflow topology and great difference in both the size of data transferred among tasks and the runtime of tasks, it is difficult or even impossible to reasonably allocate the budget to all tasks at one time before scheduling. Motivated by this, the proposed TDSA will trigger the function *DistrSubBudget*(), shown in Algorithm 2, to fairly reallocate budget for all unscheduled tasks after scheduling each batch of ready tasks. In this way, it is helpful to timely correct the fluctuation of budget usage. When allocating a sub-budget for a task, this function takes into account both the runtime of this task and the amount of data to receive.

For the function *DistrSubBudget*() in Algorithm 2, its inputs are the set of unscheduled tasks and the remaining budget so far, and the output is the updated sub-budgets for

Algorithm 2 *DistrSubBudget*()

```
Input: The set of unscheduled tasks $T_u$; remaining
       budget $remB$;
Output: Subbudgets of unscheduled tasks;
1  $W \leftarrow 0$;
2  for $t_i \in T_u$ do
3      $w(t_i) \leftarrow \sum_{t_p \in Pred(t_i)} w(e_{p,i})/m + e_i$;
4      $W \leftarrow W + w(t_i)$;
5  for $t_i \in T_u$ do
6      $b(t_i) \leftarrow \frac{w(t_i)}{W} \times remB$;
```

all the unscheduled tasks. The symbol W is initialized as zero to record the total weight of all the unscheduled tasks (line 1). For a task, its weight is defined as the sum of data receiving time and running time (line 3). After calculating the weights of all unscheduled tasks, the remaining budget is distributed fairly according to the weight ratio of these tasks (lines 5-6).

Since there is no data transmission among different tasks on the same resources, and the data constraints among tasks cause a large number of time slots on resources, then duplicating the precursors of a task to the same resource is conducive to simultaneously improving its start/completion time and monetary cost. Based on this, we design a task scheduling mechanism based on task duplication in the function *TaskSchedule*(). As shown in Algorithm 3, its inputs are: the task t_i waiting to be scheduled and its sub-budget, the available resources, and resource types. Then, the outputs of this function are: the set of updated resources, and the marginal cost of executing the task t_i.

At first, four parameters are initialized (lines 1-2), in which r_* is used to record the selected resource, *minFT* and *minC* respectively represent the minimum completion time and cost, and *Dup* represents the duplication plan of the precursors of task t_i. Then, this function tries to choose a resource that can complete task t_i under its sub-budget constraint with the minimum completion time (lines 8-11). If no such a resource is feasible, the one that takes the minimum marginal monetary cost is chosen (Line 12-14). When searching an available resource for task t_i, function *TaskSchedule*() asymptotically selects the precursor with the latest arrival time of output data (lines 15-19), and then duplicates the selected precursor to a feasible time slot of the same resource as the task t_i (lines 20-23). The above task duplication operation will be iterated until the selected precursor has been mapped to this resource (line 27) or no time slot on the resource is feasible for task t_i (line 25).

After checking all the available resources, function *TaskSchedule*() attempts to add a new resource to enhance task completion time within the sub-budget (lines 31-33), or complete this task spending as little as possible (lines 34-35). The parameter u_* is used to record the selected resource type.

Algorithm 3 *TaskSchedule*()

```
Input: A workflow task, denoted as $t_i$; sub-budget $b(t_i)$;
       the set of available resources $R$; a set of
       available resource types $U$;
Output: The set of available resources $R$; marginal cost
        $mC$ of executing $t_i$;
1  $r_* \leftarrow$NULL; $minFT \leftarrow +\infty$;
2  $minC \leftarrow +\infty$; $Dup \leftarrow \emptyset$;
3  for $r_k \in R$ do
4      $tempDup \leftarrow \emptyset$;
5      while $TRUE$ do
6          $ct_{i,k} \leftarrow$ Get finish time of $t_i$ on $r_k$;
7          $marC \leftarrow$ Get margical cost of $t_i$ on $r_k$;
8          if $marC < b(t_i)$ then
9              if $ct_{i,k} < minFT$ then
10                 $minFT \leftarrow ct_{i,k}$;
11                 $r_* \leftarrow r_k$; $Dup \leftarrow tempDup$;
12         else if $marC < minC$ then
13             $minC \leftarrow marC$;
14             $r_* \leftarrow r_k$; $Dup \leftarrow tempDup$;
15         $bp(t_i) \leftarrow$ NULL; $latAT \leftarrow 0$;
16         for $t_p \in Pred(t_i)$ do
17             $tempAT \leftarrow ct_{p,r(p)} + dt_{r(p),k}$;
18             if $tempAT > latAT$ then
19                 $bp(t_i) \leftarrow t_p$; $latAT \leftarrow tempAT$;
20         if $bp(t_i) \neq NULL \bigwedge bp(t_i)$ is not on $r_k$ then
21             $slot \leftarrow$ Find a feasible time slot for $bp(t_i)$ on
               $r_k$;
22             if $slot$ is not empty then
23                 $tempDup \leftarrow tempDup \bigcup \{(bp(t_i), slot)\}$;
24             else
25                 BREAK;
26         else
27             BREAK;
28 $u_* \leftarrow$ NULL;
29 for $u \in U$ do
30     $[ct_{i,k}, marC] \leftarrow$ Get finish time and margical cost of
       $t_i$ on a new resource with type $u$;
31     if $marC < b(t_i)$ then
32         if $ct_{i,k} < minFT$ then
33             $minFT \leftarrow ct_{i,k}$; $u_* \leftarrow u$;
34     else if $marC < minC$ then
35         $minC \leftarrow marC$; $u_* \leftarrow u$;
36 if $u_* \neq NULL$ then
37     $r^{u_*}_{|R|+1} \leftarrow$ Rent a new resource with type $u_*$;
38     $R \leftarrow R \bigcup \{r^{u_*}_{|R|+1}\}$;
39     Allocate task $t_i$ to resource $r^{u_*}_{|R|+1}$;
40 else
41     Append $t_i$ to the task queue on resource $r_*$;
42     Duplicate $t_i$'s predecessors according to $Dup$;
```

If adding a new resource is a better choice, i.e., $u_* \neq NULL$ (line 36), a resource with the selected type will be enrolled, and task t_i will be mapped to this new resource (line 39). Otherwise, the task will be mapped to the selected resource (line 41), and its precursors will be duplicated according to the duplication plan *Dup* (line 42).

To show the advantages of task duplication based workflow scheduling mechanism, the workflow in Figure 1 is taken as an example to compare the makespan and cost of executing this workflow with and without task duplication. For simplicity, assume that there is only one type of resource, denoted as 1, and its price is \$0.5/h. The main parameters of workflow, including task runtime and data transfer time among tasks, are shown in Tables 1 and 2. The symbol – in Table 2 denotes that there is no data transfer between the corresponding two tasks. The numerical value represents the data transfer time from a task to its successor (abbreviated to Succ.) when they are not on the same resource. For instance, the value 0.3 in the third column of the second row indicates the data transfer time from task t_1 to task t_2. The schedules for the workflow without and with task duplication are given in Figure 2 (a) and (b), respectively.

TABLE 1. Runtime (h) of tasks on the most powerful resource.

Task	t_1	t_2	t_3	t_4	t_5	t_6
Runtime	0.1	0.2	0.2	0.1	0.2	0.1

TABLE 2. Data transfer time (h) between tasks.

Task \ Succ.	t_1	t_2	t_3	t_4	t_5	t_6
t_1	–	0.3	0.1	–	–	–
t_2	–	–	–	0.1	0.7	–
t_3	–	–	–	–	0.2	0.5

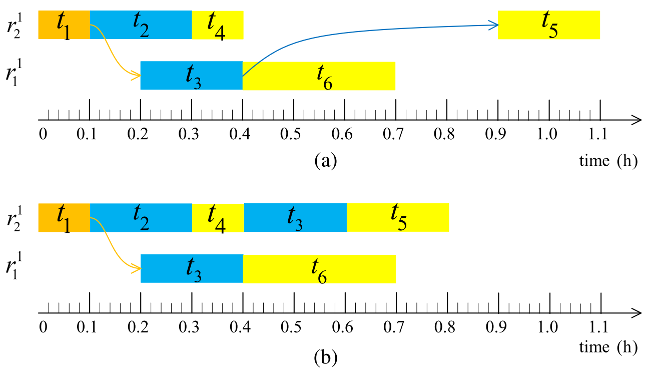

FIGURE 2. Gantt charts for workflow scheduling results.

Suppose tasks t_1, t_2, and t_4 have been scheduled to resource r_2^1, while tasks t_3 and t_6 have been scheduled to resource r_1^1. Since the tasks t_1, t_2, and t_4 are on the same resource, there is no data transmission between them, and the completion time of task t_4 is 0.4 hour. Besides, since task t_3 needs to wait for the output data of task t_2 to pass from resource r_2^1 to resource r_1^1, the start time of task t_3 is delayed to 0.2 hour. Then, the completion time of task t_6 is 0.7 hour.

If task t_5 is directly scheduled to resource r_2^1, waiting for output data of task t_3 will delay its start time to 0.9 hours. Then, the makespan of the workflow is 1.1 hours. The number of time periods for renting resources is 3, and the cost is $3 \times 0.5 = \$1.5$.

When the precursor t_3 of task t_5 is duplicated to resource r_2^1, task t_5 can directly use the output data of backup task t_3 to avoid the delay caused by data transmission. Then, the start time of task t_5 is enhanced to 0.6 hours. The makespan of the workflow is 0.8 hours. Besides, the number of time periods for renting resources is 2, and the cost is $2 \times 0.5 = \$1.0$.

Compared with the scheduling results in Figure 2 (a) and figure (b), we can see the advantages of task duplication in reducing makespan and cost of workflows.

IV. EXPERIMENTAL STUDIES

In this section, extensive comparative experiments are conducted to testify the effectiveness of our proposal, i.e., TDSA. The compared algorithms include TDSA-N-DB, GRP-HEFT [14], FBCWS [32], and EFT-MER [44].

The TDSA-N-DB is a variant of the proposed TDSA. Differing from TDSA, the TDSA-N-DB does not employ the dynamic sub-budget allocation mechanism, and just fairly allocates sub-budgets for all the workflow tasks before scheduling. By comparing TDSA with TDSA-N-DB, the effects of the proposed dynamic sub-budget allocation mechanism on the overall performance will be highlighted. The GRP-HEFT and FBCWS are two recent works on budget-constrained workflow scheduling in cloud computing, and we choose them as representatives of state-of-the-art algorithms. The EFT-MER is a popular workflow scheduling algorithm for optimizing the makespan.

Due to various workflows that will be employed to compare the above five algorithms, it is irrational to compare workflows' makespans. Another option is to normalize workflows' makespan before comparison. For a workflow, its normalized makespan is defined as follows:

$$NM(DAG) = \frac{\max_{t_i \in T}\{ct_{i,r(t_i)}\}}{\sum_{t_i \in CPT} be_i}, \tag{5}$$

where *CPT* refers to the set of tasks in the critical path of the workflow when ignoring the data transfer time among tasks, be_i denotes the minimal execution time of task t_i.

Resource utilization refers to the ratio of the time resources spend performing workflow tasks to the total time resources are used. It is also one of the key indicators to measure the performance of workflow scheduling algorithms for cloud platforms, and we also employ it to compare the performance of the five scheduling algorithms.

A. EXPERIMENT DESIGN

Referring to works [2], [45], the synthetic workflows are randomly produced according to the following attributes: 1) the number of workflow tasks; 2) tasks' execution time, denoted as *RunTimeBase* in second; 3) ratio of transmission to computing time *CCR*; 4) parallelism factor; 5) task's out-degree.

In addition, the parameter *BudgetBase* is used to control workflows' budgets, which can be computed as follows.

$$B = BudgetBase \times minCost(DAG), \tag{6}$$

where $minCost(DAG)$ denotes the cost of executing all the workflow tasks on the cheapest resources.

The resource types in cloud computing are set based on Amazon EC2 instances, six resource types are employed, as given in Table 3.

TABLE 3. Main parameters for different resource types.

Type index	Name	Price ($/h)	Compute Unit	Bandwidth (GB/s)
1	m1.small	0.059	1.70	3.90
2	m1.medium	0.120	3.74	8.52
3	m1.large	0.240	7.50	8.52
4	m1.xlarge	0.480	15.0	13.1
5	m3.2large	0.900	30.0	13.1

For each experimental setup, the six workflow scheduling algorithms are repeated 51 times independently, and the average values, upper and lower bounds are plotted.

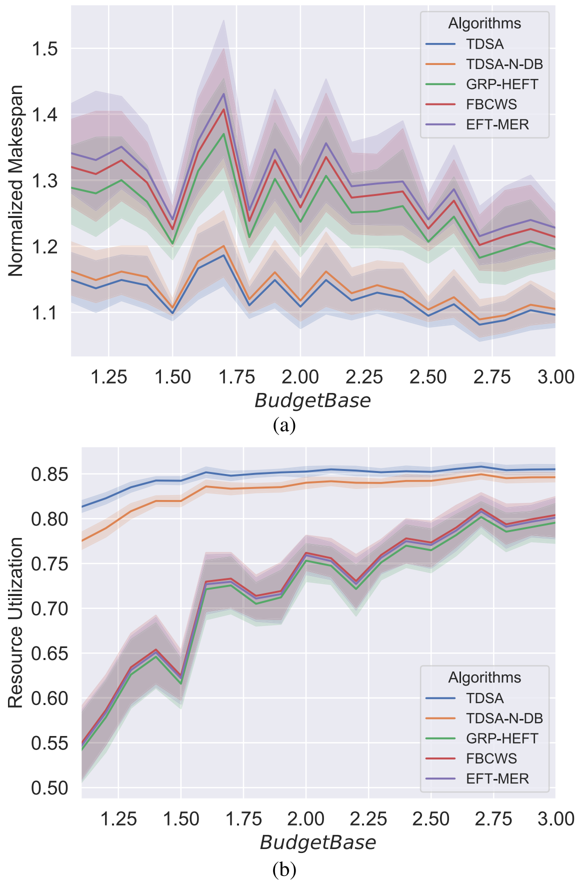

FIGURE 3. Effects of *BudgetBase*.

B. PERFORMANCE VS. BUDGET

To assess the effects of workflow budget on algorithms' performance, in this subsection, we adjust the budget control parameter *BudgetBase* from 1.1 to 3.0, and compare the performance of the proposed TDSA with the other four algorithms in terms of normalized makespan and resource utilization. The comparison results are plotted in Fig. 3.

It can be seen from Figure 3 (a) that with the increase of parameter *BudgetBase*, the normalized makespan of the five algorithms, i.e., TDSA, TDSA-N-DB, GRP-HEFT, FBCWS, and EFT-MER, shows a downward trend on the whole. This is because the larger the parameter *BudgetBase* is, the workflows will have higher budgets. Then, scheduling algorithms can select more powerful resources to perform workflow tasks. In addition, the proposed algorithm TDSA and its variant TDSA-N-DB generate schedules with much lower makespan than that of the other three existing workflow scheduling algorithms. This can be attributed to the fact that task duplication mechanism in TDSA and TDSA-N-DB can effectively advance workflow tasks' completion time and thus shorten the makespan of the entire workflow. The difference between TDSA and TDSA-N-DB is that TDSA-N-DB does not use the dynamic sub-budget allocation mechanism. It can be seen from Figure 3 (a) that TDSA performs better than TDSA-N-DB in terms of normalized makespan. These comparison results demonstrate that the proposed dynamic sub-budget allocation mechanism in this article is beneficial to shorten workflows' makespan.

As shown in Figure 3 (b), resource utilization increases with the increase of parameter *BudgetBase*. For example, when increasing parameter *BudgetBase* from 1.1 to 3.0, the resource utilization of the algorithm TDSA increases from 0.81 to 0.86. This is because with higher budgets for workflows, the choice space of each workflow task in the heterogeneous cloud environment becomes larger, which is conducive to improving resource utilization. Besides, the resource utilization of the three compared algorithms, GRP-HEFT, FBCWS, and EFT-MER, increases faster than TDSA, TDSA-N-DB. The main reason is that when the budget is low, the resource utilization of these three algorithms is very low, and their improvement space is large.

C. PERFORMANCE VS. DATA TRANSFER TIME

To examine the effects of data transfer time on the performance of the five algorithms, the parameter *CCR* is increased from 0.1 to 2.0 with the step size of 0.1 in this subsection. The normalized makespan and resource utilization of the five

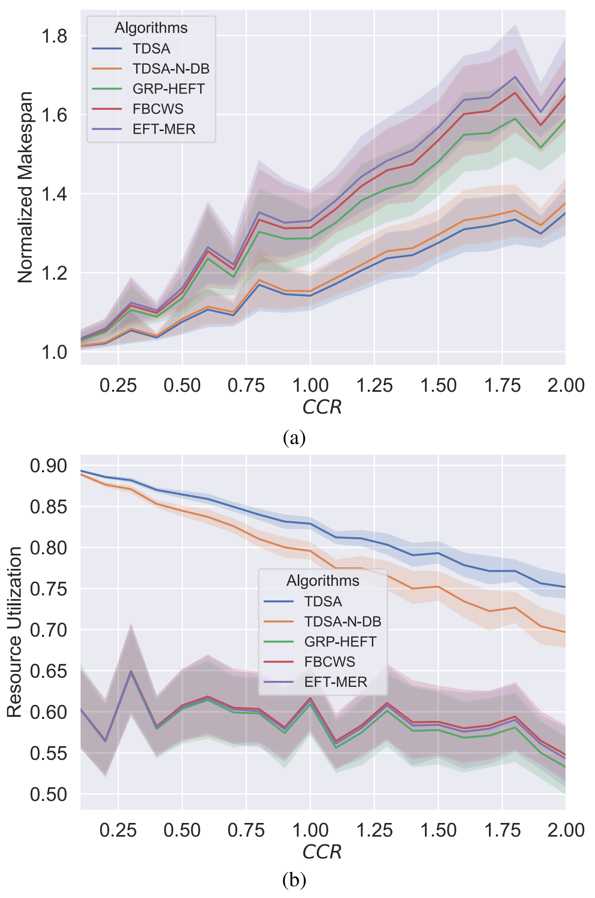

FIGURE 4. Effects of *CCR*.

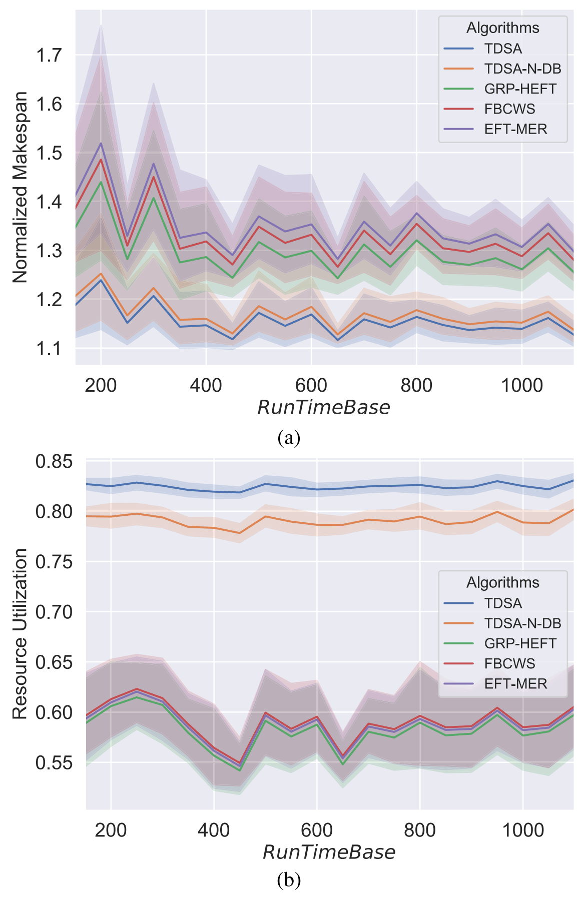

FIGURE 5. Effects of *RunTimeBase*.

algorithms, i.e., TDSA, TDSA-N-DB, GRP-HEFT, FBCWS, and EFT-MER, are provided in Fig. 4.

As shown in Figure 4 (a), workflows' normalized makespan increases with the increase of parameter *CCR*. This can be explained as the larger the parameter *CCR* is, the larger the amount of data among workflow tasks is, and the longer the data transfer time is, resulting in the longer workflows' makespan. Similar to Figure 3 (a), the normalized makespan generated by the algorithm proposed in this article is much better than that of the other three existing scheduling algorithms, and with the increase of parameter *CCR*, the gap becomes larger.

From Figure 4 (b), we can observe that resource utilization of all the five scheduling algorithms shows a downward trend. This is because, with the increase of data size among workflow tasks, the time slot between tasks will be expanded. Then, among the five scheduling algorithms, the proposed TDSA achieves the best performance overall.

D. PERFORMANCE VS. TASK RUNTIME

Task runtime is also one of the important parameters for workflows. In this section, the parameter *RunTimeBase* is increased from 150 to 1100 seconds to compare the five algorithms' performance in terms of normalized makespan and resource utilization.

TABLE 4. The main parameters of actual workflows.

Workflows	Scale	Data Size (GB)	Number of Edges	Mean Execution Time (Sec.)
Montage	medium	12.61	433	10.6
	large	129.65	4485	11.4
Inspiral	medium	273.50	319	206.1
	large	2645.95	3248	227.3
CyberShake	medium	25.49	380	31.5
	large	267.27	3988	22.7
Sipht	medium	203.04	335	175.6
	large	2054.80	3528	179.1
Epigenomics	medium	4497.73	322	3954.9
	large	45736.51	3228	3858.7

Although with the increase of *RunTimeBase*, the resource requirements of both a single workflow task and whole workflow increase, due to the scalability of cloud computing resources, the normalized makespan of the five algorithms basically remains unchanged, as shown in Figure 5 (a). In this group of comparison results, the proposed algorithm and its variant are still far superior to the other three compared algorithms.

As illustrated in Figure 5 (b), the resource utilization of the five algorithms basically remains stable with the increase of parameter *RunTimeBase*. Although the runtime

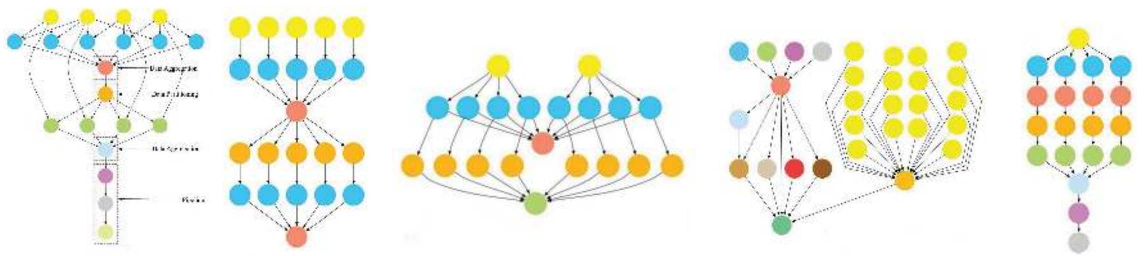
(a) Montage (b) Inspiral (c) CyberShake (d) Sipht (e) Epigenomics

FIGURE 6. Examples of actual workflows.

TABLE 5. Experimental results based on actual workflow traces.

		Montage		Inspiral		CyberShake		Sipht		Epigenomics	
		medium	large	medium	large	medium	large	medium	large	medium	large
Normalized Makespan	*TDSA*	1.7620	1.1767	1.7467	2.2281	1.1567	1.1648	1.0329	1.0314	1.5420	2.5950
	TDSA-N-DB	2.0178	1.2441	2.1275	2.4909	1.2161	1.2451	1.0644	1.1210	1.7682	3.0063
	GRP-HEFT	1.8004	1.1834	1.8520	2.3539	1.1994	1.1827	1.0421	1.0425	1.5986	3.5522
	FBCWS	1.9100	1.2196	2.5996	4.5921	1.3236	1.3277	1.0715	1.0760	5.9253	7.6515
	EFT-MER	2.2159	1.2937	3.3514	6.1119	1.4520	1.4756	1.1740	1.1581	8.0620	10.4454
Resource Utilization	*TDSA*	0.6432	0.7209	0.8036	0.7832	0.9063	0.8979	0.9303	0.9225	0.9416	0.8106
	TDSA-N-DB	0.5642	0.6033	0.7002	0.7038	0.8990	0.8877	0.8810	0.8558	0.9244	0.7780
	GRP-HEFT	0.4879	0.4563	0.6160	0.5919	0.6699	0.6167	0.6006	0.7185	0.8141	0.8094
	FBCWS	0.4354	0.3782	0.5774	0.4598	0.6692	0.6913	0.7279	0.8448	0.7788	0.7247
	EFT-MER	0.4370	0.3748	0.5065	0.3946	0.6380	0.6603	0.7265	0.8319	0.6385	0.6990

of a single task becomes longer with larger *RunTimeBase*, when the parameter *CCR* is fixed, the data transfer time among tasks also increases correspondingly, and the time slots on resources is correspondingly lengthened. In addition, for the indicator resource utilization, the proposed TDSA on average is 4.34%, 29.62%, 28.11%, and 29.07% higher than algorithms TDSA-N-DB, GRP-HEFT, FBCWS, and EFT-MER, respectively.

E. RESULTS BASED ON REAL-WORLD WORKFLOW TRACES

To assess the effectiveness of the proposed mechanisms in practical applications, the five workflow scheduling algorithms are compared in the context of five classes of workflows coming from various fields such as Montage (astronomy), Inspiral (gravitational-wave physics), CyberShake (earthquake science), Sipht (bioinformatics), and Epigenomics (DNA sequence) [46]. The topologies of workflows belonging to these five classes are shown Figure 6. From these examples, we can observe that these workflows cover various kinds of complex task dependencies.

For each type of workflow, two kinds of instances with the medium and large scale of tasks are adopted. The number of tasks is around 100 for medium scale workflows, while that is around 1000 for large scale workflow. Some relevant parameters of these workflows are summarized in Table 4. Besides, the following three main parameters of workflows are summarized: (1) the total amount of data that needs to be transferred among workflow tasks, (2) the number of edges among tasks, (3) and mean execution time of workflow tasks. In the context of these actual workflows, the comparison results are given in Table 5.

As illustrated in Table 5, the proposed TDSA is much better than the three existing scheduling algorithms, i.e., GRP-HEFT, FBCWS, and EFT-MER, in terms of both the normalized makespan and resource utilization. For the workflow Epigenomics, the superiority of the algorithm TDSA over the GRP-HEFT, FBCWS, and EFT-MER is more obvious. The main reason is that this class of workflows is data-intensive, and the data transfer time among tasks is much longer than tasks' runtime. The comparison results on workflow Epigenomics demonstrate the effectiveness of the proposed task duplication mechanism in scheduling data-intensive workflows.

V. CONCLUSION AND FUTURE WORK

In this work, we have studied the problem of workflow scheduling on heterogeneous cloud resources, and designed a task duplication based scheduling algorithm to exploit the existing time slots on resources to advance workflow tasks' start/completion time, while ensuring the budgets of workflows. Moreover, a dynamic sub-budget allocation mechanism was designed and embedded into the proposed algorithm to support accurate use of budget. At last, we assessed the proposed scheduling algorithm by comparing it with four baseline algorithms in the context of both synthetic and real-world workflows. The comparison results demonstrated the superiority of the proposed algorithm in terms of normalized makespan and resource utilization.

The running time of workflow tasks and the amount of data transferred among workflow tasks are highly uncertain. Future efforts can be dedicated to dealing with uncertainties in workflow scheduling. Evolutionary computing has shown strong search capacity in many optimization fields. How to design problem-specific evolutionary optimization mechanisms to solve workflow scheduling problems in cloud computing also worths future direction.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] H. Chen, X. Zhu, D. Qiu, L. Liu, and Z. Du, "Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 9, pp. 2674–2688, Sep. 2017.

[3] M. A. S. Netto, R. N. Calheiros, E. R. Rodrigues, R. L. F. Cunha, and R. Buyya, "HPC cloud for scientific and business applications: Taxonomy, vision, and research challenges," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 1–29, Apr. 2018.

[4] G. B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, "The application of cloud computing to astronomy: A study of cost and performance," in *Proc. 6th IEEE Int. Conf. e-Sci. Workshops*, Dec. 2010, pp. 1–7.

[5] Z. Lv and L. Qiao, "Analysis of healthcare big data," *Future Gener. Comput. Syst.*, vol. 109, pp. 103–110, Aug. 2020.

[6] J. Ekanayake, T. Gunarathne, and J. Qiu, "Cloud technologies for bioinformatics applications," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 998–1011, Jun. 2011.

[7] Y. Liu, C. Yang, and Q. Sun, "Thresholds based image extraction schemes in big data environment in intelligent traffic management," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 5, 2020, doi: 10.1109/TITS.2020.2994386.

[8] Z. Lv and W. Xiu, "Interaction of edge-cloud computing based on SDN and NFV for next generation IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5706–5712, Jul. 2020.

[9] Z. Lv and H. Song, "Mobile Internet of Things under data physical fusion technology," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4616–4624, May 2020.

[10] H. Chen, J. Wen, W. Pedrycz, and G. Wu, "Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds," *IEEE Trans. Big Data*, vol. 6, no. 1, pp. 131–144, Mar. 2020.

[11] X. Zeng, S. Garg, M. Barika, A. Y. Zomaya, L. Wang, M. Villari, D. Chen, and R. Ranjan, "SLA management for big data analytical applications in clouds: A taxonomy study," *ACM Comput. Surv.*, vol. 53, no. 3, pp. 1–40, Jul. 2020.

[12] Q. Zhu, "Research on road traffic situation awareness system based on image big data," *IEEE Intell. Syst.*, vol. 35, no. 1, pp. 18–26, Jan. 2020.

[13] J. Yan, W. Pu, S. Zhou, H. Liu, and Z. Bao, "Collaborative detection and power allocation framework for target tracking in multiple radar system," *Inf. Fusion*, vol. 55, pp. 173–183, Mar. 2020.

[14] H. R. Faragardi, M. R. Saleh Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.

[15] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.

[16] M. H. Hilman, M. A. Rodriguez, and R. Buyya, "Multiple workflows scheduling in multi-tenant distributed systems: A taxonomy and future directions," *ACM Comput. Surv.*, vol. 53, no. 1, pp. 1–39, May 2020.

[17] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, "Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 1–11, Sep. 2017.

[18] S. Singh and I. Chana, "Cloud resource provisioning: Survey, status and future research directions," *Knowl. Inf. Syst.*, vol. 49, no. 3, pp. 1005–1069, Dec. 2016.

[19] B. Cao, J. Zhao, Y. Gu, S. Fan, and P. Yang, "Security-aware industrial wireless sensor network deployment optimization," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5309–5316, Aug. 2020.

[20] B. Cao, J. Zhao, P. Yang, Y. Gu, K. Muhammad, J. J. P. C. Rodrigues, and V. H. C. de Albuquerque, "Multiobjective 3-D topology optimization of next-generation wireless data center network," *IEEE Trans. Ind. Informat.*, vol. 16, no. 5, pp. 3597–3605, May 2020.

[21] L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, "Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems," *IEEE Access*, vol. 8, pp. 205099–205110, 2020.

[22] L.-C. Canon, A. K. W. Chang, Y. Robert, and F. Vivien, "Scheduling independent stochastic tasks under deadline and budget constraints," *Int. J. High Perform. Comput. Appl.*, vol. 34, no. 2, pp. 246–264, Mar. 2020.

[23] Y. Wang and W. Shi, "Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 306–319, Jul. 2014.

[24] V. Arabnejad, K. Bubendorfer, and B. Ng, "Budget and deadline aware e-science workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 29–44, Jan. 2019.

[25] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: A survey," *J. Supercomput.*, vol. 71, no. 9, pp. 3373–3418, Sep. 2015.

[26] Z. Lv and N. Kumar, "Software defined solutions for sensors in 6G/IoE," *Comput. Commun.*, vol. 153, pp. 42–47, Mar. 2020.

[27] B. Cao, X. Wang, W. Zhang, H. Song, and Z. Lv, "A many-objective optimization model of industrial Internet of Things based on private blockchain," *IEEE Netw.*, vol. 34, no. 5, pp. 78–83, Sep. 2020.

[28] K. K. Chakravarthi and L. Shyamala, "TOPSIS inspired budget and deadline aware multi-workflow scheduling for cloud computing," *J. Syst. Archit.*, vol. 114, Mar. 2021, Art. no. 101916.

[29] H. Arabnejad, J. G. Barbosa, and R. Prodan, "Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources," *Future Gener. Comput. Syst.*, vol. 55, pp. 29–40, Feb. 2016.

[30] 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.

[31] T. Sun, C. Xiao, and X. Xu, "A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained," *Cluster Comput.*, vol. 22, no. S3, pp. 5987–5996, May 2019.

[32] N. Rizvi and D. Ramesh, "Fair budget constrained workflow scheduling approach for heterogeneous clouds," *Cluster Comput.*, vol. 23, no. 4, pp. 3185–3201, Dec. 2020.

[33] B. Cao, S. Fan, J. Zhao, P. Yang, K. Muhammad, and M. Tanveer, "Quantum-enhanced multiobjective large-scale optimization via parallelism," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100697.

[34] N. Zhou, W. Lin, W. Feng, F. Shi, and X. Pang, "Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment," *Cluster Comput.*, pp. 1–15, Sep. 2020.

[35] M. K. Arbab, M. Naghibzadeh, and S. R. K. Tabbakh, "Communication-critical task duplication for cloud workflow scheduling with time and budget concerns," in *Proc. 9th Int. Conf. Comput. Knowl. Eng. (ICCKE)*, Oct. 2019, pp. 255–262.

[36] I. Gupta, M. S. Kumar, and P. K. Jana, "Task duplication-based workflow scheduling for heterogeneous cloud environment," in *Proc. 9th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2016, pp. 1–7.

[37] M. S. Kumar, I. Gupta, and P. K. Jana, "Duplication based budget effective workflow scheduling for cloud computing," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, Springer, 2019, pp. 90–98.

[38] I. Casas, J. Taheri, R. Ranjan, L. Wang, and A. Y. Zomaya, "A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems," *Future Gener. Comput. Syst.*, vol. 74, pp. 168–178, Sep. 2017.

[39] X. Zhu, J. Wang, H. Guo, D. Zhu, L. T. Yang, and L. Liu, "Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 12, pp. 3501–3517, Dec. 2016.

[40] S. S. Mousavi Nik, M. Naghibzadeh, and Y. Sedaghat, "Task replication to improve the reliability of running workflows on the cloud," *Cluster Comput.*, to be published.

[41] A. R. Setlur, S. J. Nirmala, H. S. Singh, and S. Khoriya, "An efficient fault tolerant workflow scheduling approach using replication heuristics and checkpointing in the cloud," *J. Parallel Distrib. Comput.*, vol. 136, pp. 14–28, Feb. 2020.
[42] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds," *Future Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, Jan. 2013.
[43] H. M. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, Jun. 2013.
[44] Y. C. Lee, H. Han, A. Y. Zomaya, and M. Yousif, "Resource-efficient workflow scheduling in clouds," *Knowl.-Based Syst.*, vol. 80, pp. 153–162, May 2015.
[45] T. Xiaoyong, K. Li, Z. Zeng, and B. Veeravalli, "A novel security-driven scheduling algorithm for precedence-constrained tasks in heterogeneous distributed systems," *IEEE Trans. Comput.*, vol. 60, no. 7, pp. 1017–1029, Jul. 2011.
[46] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, Mar. 2013.

FUGUANG YAO received the B.Sc. degree in construction of engineering, the M.Sc. degree in instrumental science and technology, and the Ph.D. degree in instrumental science and technology from Chongqing University, Chongqing, China, in 2001, 2004, and 2009, respectively.

He served as the Deputy Director for the Information Center, Chongqing University of Education, where he is currently an Associate Professor. He has been engaged in research and work in the fields of cloud computing, big data, machine learning, and education informatization. He has published more than 30 academic articles. He has presided over more than ten scientific research and teaching research projects, including four provincial and ministerial projects. He holds two invention patents, three utility model patents, and three software copyrights.

CHANGJIU PU received the B.Sc. degree in computer science and technology from Chongqing Three Gorges University, Chongqing, China, in 2003, and the M.Sc. degree in computer application technology from Southwest University, Chongqing, in 2009.

He currently serves as an Associate Professor and the Director for the Data Service Center, Information Center, Chongqing University of Education. He has been engaged in research and work in the fields of cloud computing, big data, machine learning, and education informatization. He has published more than 20 academic articles. He has presided over seven scientific research and teaching research projects. He holds eight utility model patents and ten software copyrights.

ZONGYIN ZHANG received the B.Sc. degree in computer science and technical specialty from Chongqing Normal University, Chongqing, China, in 2009.

He serves as a Staff Member for the Information Center, Chongqing University of Education. He has been engaged in campus networks and education informatization.