Cluster Computing (2023) 26:1737–1751
https://doi.org/10.1007/s10586-020-03176-1

Budget-deadline constrained approach for scientific workflows scheduling in a cloud environment

Naqin Zhou[1] · Weiwei Lin[2] · Wei Feng[3] · Fang Shi[2] · Xiongwen Pang[4]

Received: 18 April 2020 / Revised: 9 July 2020 / Accepted: 21 August 2020 / Published online: 8 September 2020

Abstract
In cloud computing environments, it is a great challenge to schedule a workflow application because it is an NP-complete problem. Particularly, scheduling workflows with different Quality of Service (QoS) constraints makes the problem more complex. Several approaches have been proposed for QoS workflow scheduling, but most of them are focused on a single QoS constraint. Therefore, this paper presents a new algorithm for multi-QoS constrained workflow scheduling, cost, and time, named Budget-Deadline Constrained Workflow Scheduling (BDCWS). The algorithm builds the task optimistic available budget based on the execution cost of the task on the slowest virtual machine and the optimistic spare budget, and then builds the set of affordable virtual machines according to the task optimistic available budget to control the range of virtual machine selection, and thus effectively controls the task execution cost. Finally, a new balance factor and selection strategy are given according to the optimistic spare deadline and the optimistic spare budget, so that the execution cost and time consumption of the control task are more effective. To evaluate the proposed algorithm, we experimentally evaluated our algorithm using real-world workflow applications. The experimental results show that compared with DBWS (Deadline-Budget Workflow Scheduling) and BDAS (Budget-Deadline Aware Scheduling), the proposed algorithm has a 26.3–79.7% higher success rate. Especially when the deadline and budget are tight, the improvement is more obvious. In addition, the best cost frequency of our algorithm achieves a 98%, which is more cost-competitive than DBWS.

Keywords Cloud computing · Scientific workflow · QoS scheduling · Budget · Virtual machine

1 Introduction

The development of cloud computing technology provides a good platform for parallel applications, especially scientific workflows, such as Epigenomics in bioinformatics, Montage from astronomy and LIGO from gravitational physics. These platforms offer numbers of networked, flexible and scalable resources and services, and users pay only for what they use. However, the inherent flexibility of cloud computing platforms, while powerful, may also lead

✉ Weiwei Lin
linww@scut.edu.cn

✉ Xiongwen Pang
augepang@163.com

Naqin Zhou
439657699@qq.com

Wei Feng
doudou187230@163.com

Fang Shi
csshifang@mail.scut.edu.cn

[1] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China

[2] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

[3] Guangzhou Branch of Shanghai Yizhong Enterprise Management Consulting Co., Ltd, Shanghai, China

[4] School of Computer, South China Normal University, Guangzhou, China

to inefficient usage and high costs when inadequate scheduling and provisioning decisions are made [1, 2]. Therefore, how to effectively perform workflow scheduling in a cloud computing environment is the key to ensuring that workflow applications benefit from the cloud computing environment.

Workflow scheduling is the mapping of each task in a workflow to a suitable resource and sorting the tasks on each resource to meet certainly given metrics. As a well-known NP problem, it has been a hot research topic in the distributed computing community for many years [3], and many scheduling algorithms have been proposed. However, the scheduling algorithms proposed in the early days are mostly based on shared community environments, such as community grids. In general, the scheduling objective is to minimize the execution time of the workflow, where workflows with QoS constraints are rarely focused on.

With the development of service-oriented grids, many workflow scheduling algorithms based on QoS constraints have been proposed, such as [4–12]. However, these methods only consider a fixed number of resources and cannot be directly applied to the cloud computing environment. Because in a cloud computing environment, resources can be acquired at any time and released when they are idle. Several scheduling algorithms have been proposed for workflows with QoS constraints in cloud computing environments in recent years, but the majority of research has focused on one of cost or time, such as [3, 13–19]. However, in cloud computing, time and cost are two of the most relevant user concerns in user-defined quality of service (QoS). Considering time and cost constraints simultaneously makes scheduling problems even more challenging. A few scheduling approaches consider both cost and time constraints, such as [20, 21]. However, the literature [20] and [21] first divide the tasks in the workflow into different levels, and then assign sub-deadline to each level. The tasks of the same level have the same sub-deadline. This way of dividing deadline by level is not conducive to each task make full use of the spare deadline to balance time and cost.

Towards this, we propose a new heuristic algorithm for multi-QoS constrained workflow scheduling, cost and time, named Budget-Deadline Constrained Scheduling (BDCWS). The BDCWS algorithm introduces an optimistic spare budget and an optimistic spare deadline, and gives a new balancing factor and selection strategy based on the optimistic spare budget and the optimistic spare deadline, making the execution cost and time consumption of the control task more effective.

The main contributions of this paper include the following. (1) A multi-QoS constrained workflow scheduling algorithm is proposed to solve the workflow scheduling problem with budget and deadline constraints in cloud computing environment; (2) The definition of new balancing factors and selection strategies based on the optimistic spare deadline and optimistic spare budget expands the time and cost adjustable range, which is more conducive to balancing time and cost consumption, thereby increasing the possibility of meeting both deadlines and budget constraints; (3) Double control of the task execution cost through the set of affordable virtual machines (built based on the task optimistic available budget) and the new balance factor, not only ensuring the cost competitiveness of the scheduling result, but also ensuring the successful rate; (4) Using real workflow applications to evaluate the algorithm, the experimental results show that our algorithm achieves a 26.3–79.7% higher success rate when compared to state of the art algorithms; (5) Introducing the cost frequency to analyze the experimental results, the results show that the scheduling results generated by our scheduling algorithm have more cost-competitive.

The rest of the paper is organized as follows. After outlining the related work in Sect. 2, we describe the system scheduling model in Sect. 3. Section 4 introduces the proposed scheduling algorithm. Section 5 shows the experimental results, and Sect. 6 concludes the paper.

2 Related work

Workflow scheduling problems have been extensively studied for many years. A large range of the presented scheduling algorithms in the early days are based on the shared community environments such as community grids. Assuming that the available computation or storage resources are limited, in general, the scheduling objective is to minimize the execution time of the workflow, where workflows with QoS constraints are rarely focused on. This problem becomes more challenging when QoS parameter constraints are considered in the scheduling problems. Time, cost, energy and reliability are common QoS parameters considered in research work in this area. Yu et al. [10] proposed a QoS workflow scheduling method based on Markov decision process, which can minimize the cost while satisfying the deadline constraints given by users in the grid. Liu et al. [11] proposed a path balance algorithm to adjust the length of each path in the workflow, and proposed a cost optimization algorithm based on path balance. Sakellariou et al. [5] proposed two algorithms, LOSS and GAIN, for time-optimized and cost-constrained. They all use other heuristics to minimize one target as an initial allocation, then implement a redistribution strategy to optimize another goal and meet the user's budget constraints. Zheng et al. [6, 7] proposed the algorithm Budget-constrained Heterogeneous Earliest Finish Time (BHEFT) that optimizes the execution time of a budget-constrained workflow. Similar to BHEFT, [4] proposed Heterogeneous Budget Constrained Scheduling (HBCS) algorithm. The algorithm defines a quantity Cost Coefficient to adjust the ratio between available budget and the cheapest possibility to control the budget and minimize the execution time of the workflow application. However, the HBCS algorithm is unfair for low-priority tasks because high-priority tasks have more budget than low-priority tasks [12]. Chen et al. [12] proposed a scheduling algorithm to solve the unfairness of the HBCS algorithm, namely, Minimizing the Schedule Length using the Budget Level (MSLBL). Wu et al. [22] proposed a heuristic algorithm of PCP-B2 with cost-constrained and time-optimized. PCP-B2 uses the PCP-wise budget distribution mechanism, which balances budget among partial critical paths according to their sequential or parallel structure nature. Prodan et al. [9] proposed a general bi-criteria scheduling heuristic called dynamic constraint algorithm (DCA). DCA models the scheduling problem as an extension of the multiple-choice knapsack problem and uses dynamic programming to optimize two criterions. DCA first selects a criterion as the primary criterion and optimizes it. Then, DCA establishes a sliding constraint and optimizes the secondary criteria within the sliding constraint. Arabnejad et al. [8] proposed a deadline-budget constrained scheduling (DBCS), which defines a quality measure to balance time and cost. Sun et al. [23] proposed a scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained. The proposed approach uses sub deadline for each task to assign priority of each task.

The above methods offer valuable experience for the opportunities and challenges of workflow scheduling in a grid environment. However, there are very big differences between cloud computing environments and grid environments in resource supply and resource pricing mechanisms [13]. References [24–29] classified and described many workflow methods in the cloud computing environments. Since the scheduling constraints in this paper are time and cost, only these two QoS parameters are considered in our review of previous work. These algorithms can be divided into three categories, one is to constrain one parameter to optimize another parameter, two-parameter optimization and two-parameter constraint.

Concerning the optimization of cost constrained to time, Mao et al. [30] proposed an “auto-scaling” mechanism that automatically increases/decreases computing resources based on workload information to minimize execution costs while meeting deadlines. Abrishami et al. [31] designed two algorithms based on the Partial Critical Path (PCP) [32], which were the IaaS Cloud Partial Critical Paths (IC-PCP) one-phase algorithm and the IaaS Cloud Partial Critical Paths with Deadline Distribution (IC-PCPD2) two-phase algorithm. The purpose of the two algorithms is to minimize the cost of workflow execution while meeting a user-defined deadline. Calheiros et al. [33] gives the Enhanced IC-PCP with Replication (EIPR) algorithm for improving IC-PCP. The algorithm applies task replication to increase the chances of meeting deadlines. Rodriguez et al. [13] proposed the Particle Swarm Optimization (PSO) algorithm based on meta-heuristic optimization technology to minimize the overall workflow execution cost while meeting deadline constraints. Hong et al. [18] proposed a hybrid distribution estimation algorithm to optimize cost with deadline constraints and setup time in cloud computing. Arabnejad et al. [14] introduced a new heuristic scheduling algorithm Deadline Distribution Ratio (DDR) to solve the workflow scheduling problem with the objectives of minimizing the cost of cloud computing resources while meeting a given deadline. Anwar et al. [34] proposed a dynamic scheduling algorithm of bag of tasks based workflows to meet user-defined deadline constraints while minimizing costs. Sahni et al. [15] proposed a dynamic cost-effective deadline-constrained heuristic algorithm for scheduling scientific workflows in public clouds. Singh et al. [19] proposed a scheduling algorithm called Partition Problem based Dynamic Provisioning and Scheduling (PPDPS) to optimize the execution cost of deadline-constrained workflow applications. Meena et al. [35] proposed a meta-heuristic cost effective genetic algorithm to minimize the execution cost of workflow while meeting the deadlines of the cloud computing environment. Wu et al. [3] proposed a meta-heuristic algorithm L-ACO and a simple heuristic algorithm ProLiS to minimize the execution cost of workflow in the cloud under a deadline constraint.

Concerning the optimization of time constrained to cost, Wu et al. [36] rigorously proved that the scheduling problem of budget constraints is not only NP-complete, but also incomparable, and a heuristic solution is designed for this problem. Ghafouri et al. [16] proposed a scheduling algorithm called CB-DT (Constrained Budget-Decreased Time) to reduce makespan while meeting the budget constraints of workflow applications. In order to get a smaller makespan, the algorithm attempts to use the back-tracking method to select faster and more expensive resources for critical tasks as much as possible. Rodriguez et al. [17] proposed a budget-driven algorithm with fine-grained billing periods, the purpose of which is to optimize the way in which the budget is spent to minimize the application’s makespan (i.e., total execution time). Arabnejad et al. [37] proposed a scheduling algorithm as Budget Distribution with Trickling (BDT) and concluded that the earlier calculation of biasing the budget distribution to the workflow would result in a lower makespan within the budget. Faragardi et al. [38] proposed the Greedy Resource Provisioning and modified HEFT (GRP-HEFT) algorithm for minimizing the makespan of a given workflow subject to a budget constraint for the hourly-based cost model of modern IaaS clouds. The GRP-HEFT consists of two parts, (i) a resource provisioning algorithm and (ii) a scheduling algorithm. The resource provisioning algorithm lists the instance types according to their efficiency rate (its capacity divided by its cost). For the scheduler, [38] modified the HEFT algorithm to consider a budget limit. Rizvin et al. [39] proposed the Fair Budget-Constrained Workflow Scheduling algorithm (FBCWS) to minimize the makespan while satisfying budget constraints and a fair means of schedule for every task. Chakravarthi et al. [40] proposed the Normalization based Budget constraint Workflow Scheduling algorithm (NBWS), which controls the resource selection range of each task according to the available budget, thereby improving the probability 'best' resource selection for the task.

With regard to bi-parameter optimization, Su et al. [41] proposed a cost-effective scheduling algorithm based the concept of Pareto dominance, which produced the same makespan as the Heterogeneous Earliest Finish Time [42] (HEFT) algorithm, while significantly reducing costs. Zhu et al. [43] proposed an evolutionary multi-objective optimization (EMO) algorithm to solve the multi-objective cloud scheduling problem that minimizes makespan and cost. Choudhary et al. [44] proposed a hybrid algorithm based on Gravitational Search Algorithm (GSA) and HEFT algorithm for bi-objective workflow scheduling in cloud computing, i.e., optimization time and cost, but it is for a fixed number of virtual machines.

Regarding the bi-parameter constraint, Malawski et al. [45] proposed several dynamic (online) and static (offline) algorithms for a given budget and deadline, but these algorithms only consider one instance type. Verma et al. [46] proposed Bi-Criteria Priority based Particle Swarm Optimization (BPSO) that solves workflow scheduling problems in the cloud given deadlines and budget constraints. The algorithm can produce good scheduling results, but it has higher time complexities. In [47], Verma et al. proposed Budget and Deadline Constraint Heterogeneous Earliest Finish Time (BDHEFT) algorithm for workflow scheduling problems with deadlines and budget constraints, which is an extension of the HEFT algorithm. Similar to BDHEFT, Arabnejad et al. [20] proposed a Budget Deadline Aware Scheduling (BDAS) algorithm which solves the problem of workflow scheduling under budget and deadline constraints in the cloud. The BDAS algorithm first divides the tasks in the workflow into different levels, then allocates a budget and sub-deadline for each level, and finally selects resources for the task based on the cost time trade-off factor. The experimental results given in [20] show that the BDAS algorithm is superior to the BDHEFT algorithm. Ghasemzadeh et al. [21] proposed a Deadline-Budget Workflow Scheduling (DBWS) algorithm. Similar to the BDAS algorithm, the DBWS algorithm also divides the tasks in the workflow into different levels.

Most of the scheduling strategies mentioned above focus on constraining one QoS parameter to optimize another. In this paper, our goal is to propose a novel workflow scheduling algorithm to find a feasible solution for a workflow that meets budget and deadline constraints. To evaluate the performance of our scheduling strategy, we chose to compare with the two latest low-time complexity algorithms based on budget and deadline constraints, namely BDAS [20] and DBWS [21].

3 System scheduling model

The resource model, application model, and scheduling objective are described in this section.

The resource model, assumed in this work, consists of virtualized resources provided by an IaaS cloud service provider. The IaaS cloud service provider offers a variety of virtual machine (VM) types that can be represented as $\mathrm{VMT} = \{vmt_1, vmt_2, \ldots, vmt_{|\mathrm{VMT}|}\}$. The processing power of the various VM types can vary. The reason behind this could be (i) different MIPS rate for the virtual processor, (ii) different number of virtual cores, (iii) different memory size and storage capacity, and different storage access time. Cloud providers specify the processing power of different types of the provided VMs use a metric names Compute Unit (CU) (e.g., ECU used by Amazon EC2) [38]. $CU(vmt_k)$ denotes the compute unit of vmt_k.

In the same way, as in [20, 21], all VMs are assumed to be in the same data center or region and average bandwidth between virtual machines is roughly equal. Suppose there is no limit to the number of VM instances lease (used) by an application. Also, when a VM is leased, it requires an initial boot time for its proper initialization before it is made available to the user. Similarly, when the VM is released, it again takes some time to shut down properly. In addition, the pricing model is based on a pay-as-you-go billing scheme and the users are charged for the number of time intervals they use (lease) a VM. We use the fine-grained billing period provided by most providers, such as Amazon's EC2 cloud [48], Google Compute Engine [49] and Microsoft Azure Microsoft [50], which are 1 s billing period. The fine-grained billing period is an emerging billing period in the cloud computing environment in the past two years. Flexibility is limited when coarse-grained billing periods, so that it is easy to cause inevitable waste [17]. In September 2017, Amazon announced that EC2 will use the per-second billing period from October 2017 [51].

To ensure user benefits and increase competitiveness, many cloud providers have also pushed out fine-grained billing periods (billing per second), such as Google Compute Engine [49] and Microsoft Azure [50].

Since internal data transfer is free in most cloud environments, the data transfer cost is assumed to be zero.

3.1 Application model

A typical workflow application can generally be described as a Directed Acyclic Graph (DAG). A DAG can be modeled by a two-tuple $G = (T, E)$, where $T = \{t_1, t_2, ..., t_n\}$ is the set of nodes with each element representing a workflow task, and E is the set of directed edges with each element representing the dependency between two task nodes, that is, for $\forall e_{i,j} \in E$, task t_i must finish its execution and transfer the resulting data to solve the data dependency before task t_j starts, thus, t_i is the immediate predecessor of t_j while t_j the immediate successor of t_i.

In a given DAG, a task with no predecessor is called an entry task, and a task with no successor is called an exit task. It is assumed that a DAG has only one entry task and one exit task. If there is more than one entry task (exit task) in a DAG, a dummy entry task (exit task) with 0 weight and 0 communication can be added to the graph.

3.2 Scheduling objective

The objective of our algorithm is to find a feasible schedule for the workflow that can meet the user-defined QoS constraints, i.e., the finish time of the workflow must not exceed the user-defined time constraint (*DEADLINE*), and the total cost must not be higher than the user-defined cost constraint (*BUDGET*). For this workflow, if a schedule satisfies its time and cost constraints, the schedule is considered as a successful schedule; otherwise, it is considered as a failure.

4 Algorithm design

In this section, first, define the basic definitions that need to be used in the proposed algorithm (Sect. 4.1). Table 1 summarizes the common notations used throughout this paper.

4.1 Basic definition

Definition 1 The execution time of task t_i on virtual machine vm_k is represented by $ET_{t_i}^{vm_k}$. For workflows, Juve et al. [52] has published the execution time of tasks on Xeon@2.33 GHz CPUs (CU = 8). The execution time of tasks is reversely proportional to the CU value [38]. In the same way, as in [38], we use the execution time of tasks for a VM with $CU = 1$ as the reference execution time of tasks and denoted by ref_time. Accordingly, $ET_{t_i}^{vm_k}$ is.

$$ET_{t_i}^{vm_k} = \frac{ref\_time(t_i)}{CU(vmt(vm_k))} \tag{1}$$

Definition 2 The data transfer time between the tasks t_i and t_j is defined as:

$$TT_{i,j} = \begin{cases} data_{i,j}/bw & vm(t_i) \neq vm(t_j) \\ 0, & vm(t_i) = vm(t_j) \end{cases} \tag{2}$$

where $data_{i,j}$ is the amount of data elements that t_i sends to t_j, bw is the communication bandwidth between VMs. If t_i and t_j are assigned to the same VM (denoted by $vm(t_i) = vm(t_j)$), $TT_{i,j}$ becomes 0.

Definition 3 All immediate predecessors of task t_i are defined as:

$$pred(t_i) = \{t_j | (t_j, t_i) \in E\} \tag{3}$$

Definition 4 All immediate successors of task t_i are defined as:

$$succ(t_i) = \{t_j | (t_i, t_j) \in E\} \tag{4}$$

Definition 5 $sched_{vm_k}$ denotes a set of tasks that are scheduled to be on vm_k.

$$sched_{vm_k} = \{t_i | vm(t_i) = vm_k\} \tag{5}$$

where $vm(t_i)$ is the VM which is assigned to the task t_i.

Definition 6 The start time of the task t_i on the VM vm_k is calculated as follows:

$$ST_{t_i}^{vm_k} = \max\left\{avail_{vm_k}, \max_{t_j \in pred(t_i)} \{FT_{t_j} + TT_{i,j}\}\right\} \tag{6}$$

where $avail_{vm_k}$ is the available time of vm_k.

$$avail_{vm_k} = \begin{cases} boot\_time_{vm_k}, & sched_{vm_k} = \phi \\ FT_{t_L}^{vm_k}, & sched_{vm_k} \neq \phi \end{cases} \tag{7}$$

where $boot\_time_{vm_k}$ is the VM startup/boot time, and t_L is the last task in the sorted tasks scheduled list for the virtual machine $vm_k(sched_{vm_k})$.

Definition 7 The Finish Time of the task t_i on vm_k is calculated as follows:

$$FT_{t_i}^{vm_k} = ST_{t_i}^{vm_k} + ET_{t_i}^{vm_k} \tag{8}$$

Table 1 List of notations

Symbol	Definition
$DAG_{makespan}$	The overall schedule length for a workflow
DAG_{cost}	The total cost for executing a workflow
$ET_{t_i}^{vm_k}, ET_{t_i}^{vmt_k}$	The execution time of task t_i on VM vm_k, VM type vmt_k
$ET_{t_i}^{\min}$	The minimum execution time for task t_i among all VM type
$ET_{t_{curr}}^{\max}$	The maximum execution time for the current task t_{curr} among all VM type
$TT_{i,j}$	The transmission time from task t_i to task t_j
$pred(t_i)$	All predecessors of task t_i
$succ(t_i)$	All successors of task t_i
$vm(t_i)$	The VM where task t_i is assigned
$sched_{vm_k}$	The set of scheduled tasks on VM vm_k
$ST_{t_i}^{vm_k}$	The start time of the task t_i on VM vm_k
$ST_{t_{curr}}^{best}$	The best start time of the current task t_{curr}
FT_{t_j}	The finish time of task t_j
$FT_{t_i}^{vm_k}$	The finish time of task t_i on VM vm_k
FT_{exit}	The finish time of the exit task of the workflow
$boot\_time_{vm_k}$	The startup/boot time of VM vm_k
$Cost_{t_i}^{vm_k}$	The execution cost of task t_i on vm_k
$Cost_{t_i}^{\min}$	The minimum execution cost of task t_i among all VM type
t_L	The last task in the sorted tasks scheduled list for a VM
$vmt(vm_k)$	The VM type of VM vm_k
$CU(vmt_k)$	The compute unit of a VM type vmt_k
$VMC(vmt_k)$	The monetary cost per charging unit (price) of a VM type vmt_k
$BUDGET$	User defined budget for executing the workflow
$DEADLINE$	User defined deadline for executing the workflow

Definition 8 $DAG_{makespan}$ indicates makespan or schedule length, which is the finish time of the exit task of the workflow:

$$DAG_{makespan} = FT_{exit} \tag{9}$$

Definition 9 The execution cost of task t_i on vm_k is calculated as follows:

$$\mathrm{Cost}_{t_i}^{vm_k} = \begin{cases} \left\lceil \dfrac{FT_{t_i}^{vm_k} - ST_{t_i}^{vm_k}}{IT} \right\rceil * VMC(vmt(vm_k)), & sched_{vm_k} = \phi \\ 0, & FT_{t_i}^{vm_k} \le RT_{vm_k} \, and \, sched_{vm_k} \ne \phi \\ \left\lceil \dfrac{FT_{t_i}^{vm_k} - RT_{vm_k}}{IT} \right\rceil * VMC(vmt(vm_k)), & otherwise \end{cases} \tag{10}$$

where $VMC(vmt(vm_k))$ is the monetary cost per charging unit (price) of a VM type $vmt(vm_k)$, IT is the billing period, and RT_{vm_k} is indicated as the last interval used by last scheduled task on vm_k.

$$RT_{vm_k} = \left\lceil \frac{FT_{t_L}^{vm_k}}{IT} \right\rceil * IT \tag{11}$$

where t_L is the last task in the sorted tasks scheduled list for vm_k.

Definition 10 DAG_{cost} represents the overall cost of executing a workflow application and is defined as:

$$DAG_{\mathrm{cost}} = \sum_{t_i \in T} \{ \mathrm{Cost}_{t_i}^{vm_k} | t_i \in sched_{vm_k} \} \tag{12}$$

Definition 11 $ST_{t_{curr}}^{best}$ is the best start time of the task t_{curr}, that is, the start time of the task t_{curr} in the VM with the earliest finish time in all tested VMs.

4.2 The MW-HBDCS algorithm

The BDCWS algorithm consists of two main phases, the task selection phase and the VM selection phase.

4.2.1 Task selection phase

Tasks are selected according to their priorities. To assign a priority for a task in the DAG, the upward rank (*urank*) is computed. This *urank* represents, for a task t_i, the length of the longest path from t_i to the exit task t_{exit}, including the execution time of task t_i and it is given by Eq. (13).

$$urank_{t_i} = ET_{t_i}^{\min} + \max_{t_{child \in succ(t_i)}} \{urank_{t_{child}} + TT_{i,child}\} \tag{13}$$

where $ET_{t_i}^{\min}$ is the minimum execution time for task t_i among all VM type. For the exit node, $urank_{t_{exit}} = ET_{t_{exit}}^{\min}$.

The task with the highest *urank* value receives the highest priority, followed by the task with the next highest *urank* value, and so on.

4.2.2 VM selection phase

The VM selection phase is responsible for selecting the appropriate VM for the current task t_{curr}. To control the time and cost of consumption during the scheduling process, a limit value for each factor is needed. We define two variables, TOD (Task Optimistic Deadline) and TOAB (Task Optimistic Available Budget) as limits for time and cost, as shown in Eqs. (14) and (16), respectively.

$$TOD_{t_{curr}} = ST_{t_{curr}}^{best} + OSD_{t_{curr}} + ET_{t_{curr}}^{\min} \tag{14}$$

where $ET_{t_{curr}}^{\min}$ is the minimum execution time for task t_{curr} among all VM type, and $OSD_{t_{curr}}$ denotes the optimistic spare deadline the current task t_{curr}, as shown in Eq. (15).

$$OSD_{t_{curr}} = DEADLINE - ST_{t_{curr}}^{best} - urank_{t_{curr}} \tag{15}$$

where $urank_{t_{curr}}$ is the *urank* for the current task t_{curr}.

$$TOAB_{t_{curr}} = \frac{ET_{t_{curr}}^{\max}}{IT} * VMC(vmt(vm_{\max})) + OSB \tag{16}$$

where $ET_{t_{curr}}^{\max}$ is the maximum execution time for the current task t_{curr} among all VM type. *OSB* denotes the optimistic spare budget, which is equal to the difference of the budget and the sum of the cost for allocated tasks and the cheapest cost for unscheduled tasks. The formula is as follows:

$$OSB = BUDGET - \left[\sum_{t_i \in T_{assigned}} \mathrm{Cost}_{t_i}^{vm_{sel}} + \sum_{t_i \in T - T_{assigned}} \mathrm{Cost}_{t_i}^{\min}\right] \tag{17}$$

where vm_{sel} is the VM selected to run the scheduled task, $Cost_{t_i}^{\min}$ denotes the minimum execution cost of task t_i among all VM type, and $T_{assigned}$ is the set of allocated tasks.

In the VM selection, to guarantee that the workflow can be executed without exceeding the budget constraint, first all the $vm_j \in R$ are filtered by $TOAB_{t_{curr}}$ to construct an affordable set $S_{affordable}(t_{curr})$. R represents the set of resources (VM instances) used in previous steps of scheduling.

$$S_{affordable}(t_{curr}) = \{vm_j | \mathrm{Cost}_{t_{curr}}^{vm_j} \le TOAB_{t_{curr}}, vm_j \in \mathrm{R}\} \tag{18}$$

Then, the VM is selected using the following selection rules:

(1) If $S_{affordable}(t_{curr}) = \phi$ and $TOD_{t_{curr}} \le 0$, the VM with the earliest finish time in $R \cup R'$ is selected to execute the current task t_{curr}. R' is defined as the set of one temporary VM from each available VMT.

(2) If $S_{affordable}(t_{curr}) = \phi$ and $TOD_{t_{curr}} > 0$, the cheapest VM in $R \cup R'$ is selected to execute the current task t_{curr}.

(3) If $S_{affordable}(t_{curr}) \ne \phi$, first, filter all $vm_j \in R'$ with $TOAB_{t_{curr}}$ and build an affordable set (as in Eq. 19). Then for all $vm_j \in S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr})$ and $FT_{t_{curr}}^{vm_j} < TOD_{t_{curr}}$, calculate their Bi-factor (BF) values (as in Eq. 20). Finally, the VM is selected for the current task t_{curr} according to (a) and (b) below.

(a) If $\exists vm_j \in S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr})$ is such that $FT_{t_{curr}}^{vm_j} < TOD_{t_{curr}}$, select the VM with the smallest BF value for the current task t_{curr}.

(b) Otherwise, select the VM with the earliest finish time from $S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr})$ to execute the current task t_{curr}.

$$S'_{affordable}(t_{curr}) = \{vm_j | \mathrm{Cost}_{t_{curr}}^{vm_j} \le TOAB_{t_{curr}}, vm_j \in R'\} \tag{19}$$

$$BF_{t_{curr}}^{vm_j} = TF_{t_{curr}}^{vm_j} + CF_{t_{curr}}^{vm_j} \tag{20}$$

where $TF_{t_{curr}}^{vm_j}$ and $CF_{t_{curr}}^{vm_j}$ represent the time factor and cost factor, respectively, such as (21) and (22).

$$TF_{t_{curr}}^{vm_j} = FT_{t_{curr}}^{vm_j} / \left(\frac{ET_{t_{curr}}^{vm_j}}{urank_{t_{curr}}} * OSD_{t_{curr}}\right) \tag{21}$$

$$CF_{t_{curr}}^{vm_j} = \frac{cost_{t_{curr}}^{vm_j}}{OSB} \tag{22}$$

The pseudo code of BDCWS is shown in Table 2.

In the algorithm, *urank* value of all the tasks are first calculated by line 1. Second, in the while loop in lines 2–35, the algorithm tries to allocate VMs for all the tasks. At every loop iteration, line 3 selects the task with the highest *urank* as the current task t_{curr} In lines 4–5, its $TOD_{t_{curr}}$ and

$TOAB_{t_{curr}}$ values are calculated. Line 6 constructs a set of affordable resources $S_{affordable}(v_{curr})$ for the current task t_{curr} then, the algorithm selects a VM for the current task t_{curr} according to the defined selection rules implemented in lines 7–32. Finally, line 33 adds the mapping of the current task t_{curr} to vm_{sel} to the MAP.

The time complexities of each step in the algorithm are as follows:

a. The complexity of calculating the *urank* for all tasks is $O(e+n)$, where e is the number of edges in the DAG and n is the number of nodes.
b. At each step of the VM selection phase, the complexity of calculating the Task Optimistic Available Budget *TOAB* is $O(n \cdot m)$, where m is the number of available VM.
c. At each step of the VM selection phase, the complexities of calculating the Task Optimistic Deadline *TOD* and constructing $S_{affordable}(v_{curr})$ are each $O(m)$.

Thus, the total time is $O(e+n+n \cdot (n \cdot m+m+m))$, resulting in a total algorithm complexity of $O(n^2 \cdot m)$.

5 Experimental evaluation

Since it is very difficult to perform repeatable experiments on real datacenters, we implement our proposed approaches on the simulation platform of [3], which uses Java 2 Standard Edition V1.7.0 and is available at https://github.com/wuquanwang/workflow. In order to better evaluate the proposed algorithm, the DBWS [21] algorithm and the BDAS algorithm [20] which both are the latest two heuristic scheduling algorithms for deadline and budget constraints are chosen as the compared metrics of BDCWS algorithm.

In the experiment, we used 9 different types of VMs in [3], and each with different processing power and cost, the details are shown in Table 3. Especially, the average bandwidth between VMs is set to 20 Mbps, which is the approximate average bandwidth between computing services in Amazon EC2 [53]. The billing period and service startup time are set to 1 s and 97 s [54], respectively.

In addition, we use the same approach to calculate the cost execution of each task (Eq. 9) for all algorithms.

5.1 Budget and deadline constraints

It is necessary to define a predetermined deadline and budget constraint for each workflow to evaluate the proposed algorithm, such as Eqs. (23) and (24).

$$DEADLINE = \min_D *(0.8+\alpha_D) \tag{23}$$

$$BUDGET = \min_B *(0.8+\alpha_B) \tag{24}$$

where $\min_D$ represents makespan for scheduling target workflows using HEFT on the fastest virtual machine set, $\min_B$ represents the total execution cost of scheduling target workflows using HEFT on the cheapest-cost virtual machine set, α_D and α_B represent the time parameter and cost parameter, respectively.

In the experiments, we let a_D = [0.2,0.4,0.6, 0.8,1,2,3,4] and a_B = [0.2,0.4,0.6,0.8,1,2,3,4].

5.2 Performance metrics

To evaluate our algorithm with other algorithms, the different performance metrics are exploited, and the details are shown in the following contents.

Planning Successful Rate (PSR): it is defined as the planning success rate of finding a feasible schedule while satisfying the user-defined deadline and budget, as expressed by Eq. (25):

$$PSR = 100 \times \frac{\substack{\textit{Number of experiments that}\\ \textit{successfully meet deadline and budget}}}{\textit{Total number of experiments}} \tag{25}$$

Makespan to Deadline Ratio (MDR): the ratio of makespan achieved and deadline defined for each workflow, the expression is given by Eq. (26):

$$MDR = \frac{DAG_{makespan}}{DEADLINE} \tag{26}$$

Cost to Budget Ratio (CBR): the ratio of execution cost of the schedule produced and budget defined for each workflow, as expressed by Eq. (27):

$$CBR = \frac{DAG_{cost}}{BUDGET} \tag{27}$$

Cost frequency of algorithm A relative to algorithm B: the cost frequency includes the best cost frequency, the worse cost frequency and the equal cost frequency, and the calculation formulas are given by Eqs. (28), (29) and (30), respectively.

$$\textit{Best cost frequency} = \frac{\textit{Number of best cost}}{\textit{Total number of experiments}} \tag{28}$$

$$\textit{Worse cost frequency} = \frac{\textit{Number of worse cost}}{\textit{Total number of experiments}} \tag{29}$$

Table 2 The BDCWS algorithm

```
Input: DAG $G$ with budget $BUDGET$ and deadline $DEADLINE$
       schedule map set MAP = $\phi$
Output: schedule map set MAP
1.  Compute the urank (as defined in Eq.(13)) for each task
2.  while there is an unscheduled task do
3.      $t_{curr}$ = the next ready task with highest urank value
4.      Compute the time bound $TOD_{t_{curr}}$ using Eq.(14)
5.      Compute the cost bound $TOAB_{t_{curr}}$ using Eq.(16)
6.      Construct the set of affordable resources $S_{affordable}(t_{curr})$ using Eq.(18)
7.      if $S_{affordable}(t_{curr})$ is empty then
8.          if $TOD_{t_{curr}} \leq 0$ then
9.              for all $vm_j \in R \cup R'$ do
10.                 Calculate the finish time( $FT_{t_{curr}}^{vm_j}$ ) for task $t_{curr}$ on the VM $vm_j$
11.             end for
12.             $vm_{sel}$ = VM $vm_j$ with the smallest $FT_{t_{curr}}^{vm_j}$
13.         else
14.             for all $vm_j \in R \cup R'$ do
15.                 Calculate the execution cost（$Cost_{t_{curr}}^{vm_j}$）for task $t_{curr}$ on the VM $vm_j$
16.             end for
17.             $vm_{sel}$ = VM $vm_j$ with the cheapest $Cost_{t_{curr}}^{vm_j}$
18.         end if
19.     else
20.         Construct the set of affordable resources $S'_{affordable}(t_{curr})$ using Eq.(19)
21.         for all $vm_j \in S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr}), FT_{t_{curr}}^{vm_j} < TOD_{t_{curr}}$
22.             Calculate $BF_{t_{curr}}^{vm_j}$ using Eq.(19)
23.         end for
24.         if $\exists vm_j \in S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr}), FT_{t_{curr}}^{vm_j} < TOD_{t_{curr}}$
25.             $vm_{sel}$ = VM $vm_j$ with the smallest $BF$
26.         else
27.             for all $vm_j \in S_{affordable}(t_{curr}) \cup S'_{affordable}(t_{curr})$ do
28.                 Calculate the finish time( $FT_{t_{curr}}^{vm_j}$ ) for task $t_{curr}$ on the VM $vm_j$
29.             end for
30.             $vm_{sel}$ = VM $vm_j$ with the smallest $FT_{t_{curr}}^{vm_j}$
31.         endif
32.     end if
33.     MAP = MAP $\cup$ {Assign current task $t_{curr}$ to VM $vm_{sel}$ }
34.     Update $R = \{R \cup vm_{sel} \mid vm_{sel} \notin R\}$
35. end while
36. return MAP
```

Table 3 Capabilities and costs of available service types

Type	Compute unit	Cost ¢/s	Type	Compute unit	Cost ¢/s
$type_1$	1.0	0.12	$type_6$	3.5	0.595
$type_2$	1.5	0.195	$type_7$	4.0	0.72
$type_3$	2.0	0.28	$type_8$	4.5	0.855
$type_4$	2.5	0.375	$type_9$	5.0	1.0
$type_5$	3.0	0.48			

$$Equal\,cost\,frequency = \frac{Number\;of\;equal\;cost}{Total\;number\;of\;experiments} \tag{30}$$

where the best cost means that the cost of executing the same workflow according to the scheduling scheme generated by Algorithm A is smaller than Algorithm B, the worse cost means that the cost of executing the same workflow according to the scheduling scheme generated by Algorithm A is larger than Algorithm B, and the equal cost means that the cost of executing the same workflow according to the scheduling scheme generated by Algorithm A is the same as Algorithm B. And if the best cost frequency of algorithm A relative to algorithm B is greater than the worse cost frequency, that means algorithm A is more cost effective than algorithm B.

5.3 Experimental data sets

In our experiments, we use Workflow Generator [55] to generate three different areas of workflow, namely Epigenomics from bioinformatics, Montage from astronomy and LIGO from gravitational physics, and the details about them can be found in Ref. [52]. These workflow applications differ in terms of computational characteristics, structure, and communication data. The workflow for each type application of small size is shown in Fig. 1.

Especially, 8 type sizes are selected for each workflow application, with task sizes of 30, 50, 100, 200, 300, 400, 500, and 1000, respectively. And since 100 different workflows are tested for each size, the total number of experimental workflows is 3 × 8 × 100 = 2400.

5.4 Results and analysis

For our experiments, we define 8 different deadline factors and 8 different budget factors. Permuting both factors yield 64 different cases per workflow and each workflow includes 51,200 test cases.

5.4.1 EPIGENOMIC

As shown in Figs. 2, 3, 4, and 5, the Figs. show the results of Epigenomic, where Figs. 2 and 4 show CBR and MDR obtained by each algorithm. And the CBR and MDR values are divided into two main categories, where a valid schedule is represented by yellow and an invalid schedule is represented by green. A value is less than 1 for the CBR metric (Eq. 27) means that the algorithm meets the user defined budget. But a value CBR > 1 means that the algorithm failed to find a schedule map with the cost is lower than the user-defined budget. Similarly, the same explanation also can be applied to MDR (Eq. 26). For BDCWS, it almost uses the entire deadline and performs workflow execution at a lower cost than the defined budget to satisfy the users. As shown in Fig. 2, for all range of deadline factor α_B, the execution cost of the schedule map obtained by DBWS almost meets user-defined budget constraints. As can be found in Figs. 3 and 5, DBWS and BDAS both observed the worst performance in the first deadline factor (Fig. 3) and the first budget factor (Fig. 5), which means 100% failure.

5.4.2 MONTAGE

Figures 6, 7, 8, and 9 shows the results of MONTAGE. And from Fig. 9, it can find that the PSR obtained by the BDCWS algorithm is better than the DBWS algorithm and the BDAS algorithm, especially when the budget factor is 0.2. And the PSR obtained from the DBWS and BDAS algorithms is less than 4%, but under the same conditions, the PSR obtained from the proposed BDCWS algorithm is higher than 80%. As shown in Fig. 6, the behavior of the DBWS algorithm in MONTAGE is similar to EPIGENOMIC, that is, the costs obtained for all range of deadline factor α_D almost satisfy the budget constraint. Furthermore, an interesting feature is observed when the deadline factor α_D is equal to 0.2. As can be seen, all cost budget ratio results in a valid schedule with a success rate of 0%. It means that 100% of the failures in the algorithm are due to violating deadline.

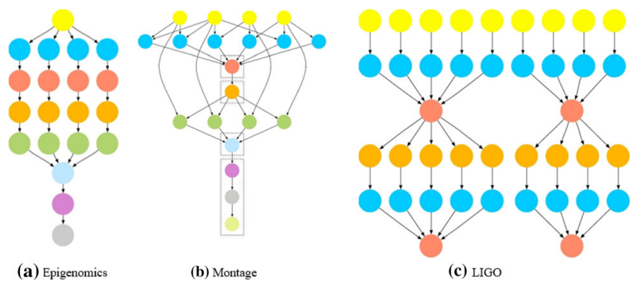


Fig. 1 The structure of the small size workflows

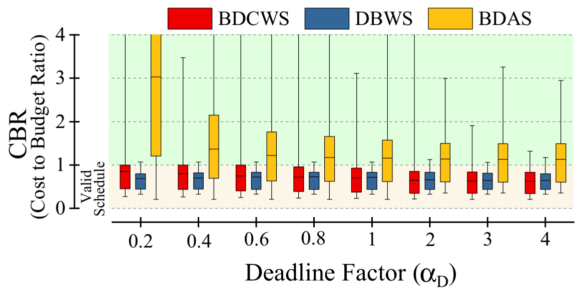


Fig. 2 CBR value for EPIGENOMIC grouped by α_D

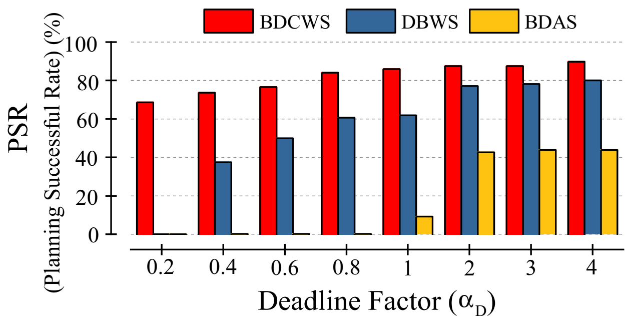


Fig. 3 PSR value for EPIGENOMIC grouped by α_D

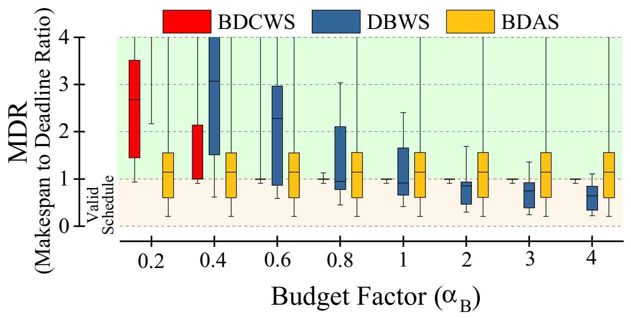


Fig. 4 MDR value for EPIGENOMIC grouped by α_B

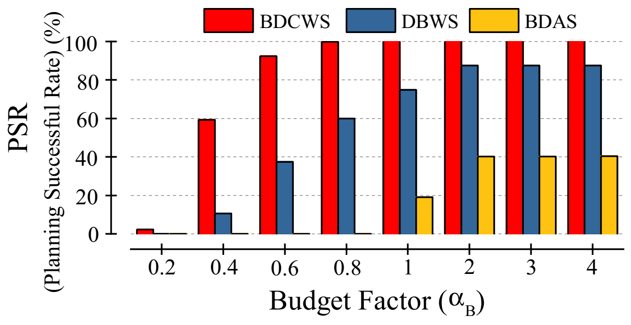


Fig. 5 PSR value for EPIGENOMIC grouped by α_B

5.4.3 LIGO

The results of LIGO are shown in Figs. 10, 11, and 12. As can be found in Fig. 12, the reason why the BDAS box plot

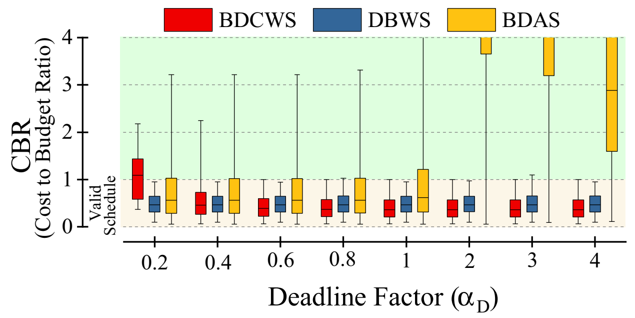


Fig. 6 MDR value for MONTAGE grouped by α_D

is not shown is any ratio greater than 1 means that a valid schedule cannot be generated, so we only display values of up to 4 to explain the results of valid schedules in more detail. Meanwhile, the behaves of BDCWS in LIGO is similar to EPIGENOMIC, it uses almost all deadlines and completes the execution of the workflow within the specified budget. And for the DBWS Algorithm, we also can find that as the budget factor increases, the ratio of makespan and deadline gradually decreases. Furthermore, it can be observed in Figs. 11 and 13 that the BDAS algorithm achieves almost 100% failure when deadlines and costs are tight.

5.4.4 The total success rate

Table 4 shows the total success rate of each algorithm scheduling workflow with different deadline factor α_D and budget factor α_B. Especially, for each workflow, we use 8 different size workflows and 100 different workflows were tested for each size. Meanwhile, for each workflow, we tested 64 different states, which are combinations of 8 different deadline factors α_D and 8 budget factors α_B. Therefore, in this study, we provided 51,200 different test cases for each scientific workflow.

It can be clearly seen from the results in Table 4 that the best performance is the BDCWS algorithm, which has a success rate of more than 81.7% in all data sets and the best performance in MONTAGE, reaching 90.8%. And the worst performer was the BDAS algorithm, especially with 99.4% of failed test cases in MONTAGE. Therefore, according to the success rate shown in Table 4, it is shown that our method is more likely to generate an acceptable scheduling scheme under defined constraints.

5.4.5 Cost frequency

Since it is meaningless to compare the cost frequency when the budget constraint and the deadline are not met, the experimental results of $a_D = [1, 2, 3, 4]$ and $a_B =$

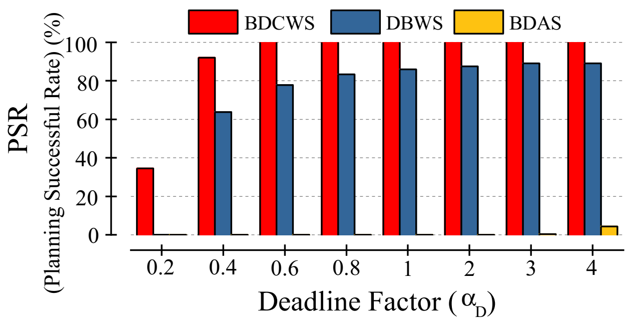


Fig. 7 PSR value for MONTAGE grouped by α_D

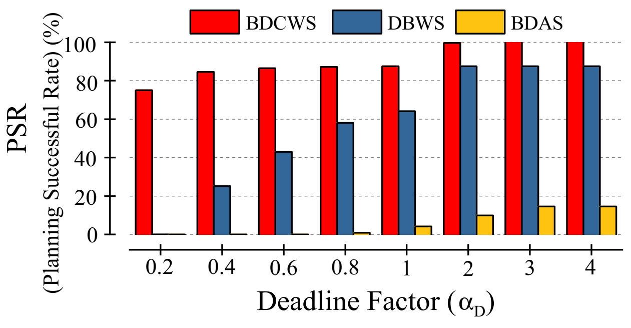


Fig. 11 PSR value for LIGO grouped by α_D

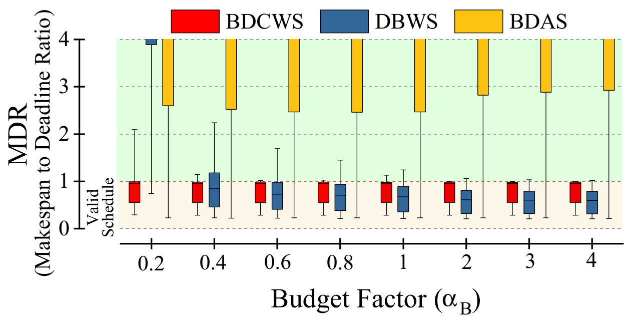


Fig. 8 MDR value for MONTAGE grouped by α_B

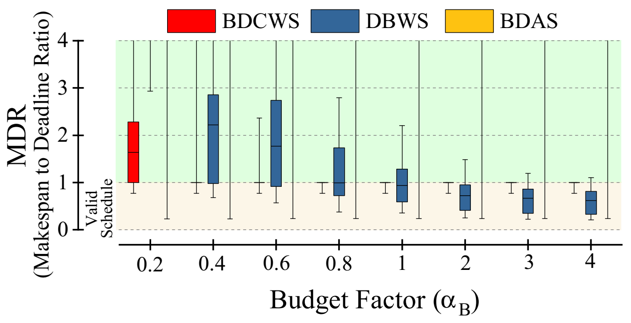


Fig. 12 MDR value for LIGO grouped by α_B

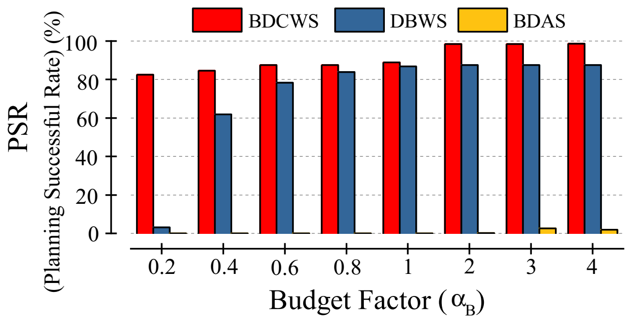


Fig. 9 PSR value for MONTAGE grouped by α_B

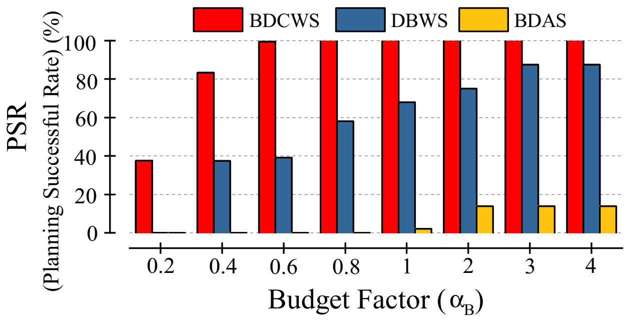


Fig. 13 PSR value for LIGO grouped by α_B

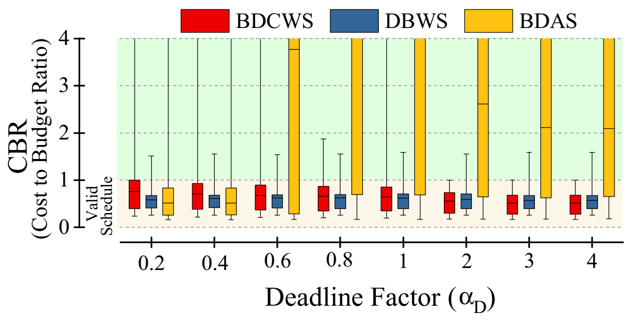


Fig. 10 MDR value for LIGO grouped by α_D

Table 4 Total success rate for three different scientific workflows

	BDCWS (%)	DBWS (%)	BDAS (%)
EPIGENOMICS	81.7	55.7	17.5
MONTAGE	90.8	72.1	0.6
LIGO	90.1	56.6	5.5
Mean	87.5	61.5	7.9

[1, 2, 3, 4][1, 2, 3, 4] are taken to compare the cost frequency, namely the success rate of the algorithms BDCWS and DWBS is reaching 100%.

Table 5 compares the best frequencies, worse frequencies, and equal cost frequencies of the algorithms BDCWS and DWBS. Compared with the DWBS algorithm, the BDCWS algorithm has the best cost frequency more than the worse frequency, especially in the MONTAGE, the best

Table 5 Cost frequency

	Best cost frequency (%)	Worse cost frequency (%)	Equal cost frequency (%)
EPIGENOMICS	98.2	1.7	0.1
MONTAGE	100	0	0
LIGO	95.8	4	0.2
Mean	98.0	1.9	0.1

cost frequency reaches 100%. Therefore, we can say that the BDCWS algorithm is more cost-competitive than the DWBS algorithm.

6 Conclusions and future work

In this paper, for the workflow scheduling problem in the cloud environment, a BDCWS algorithm is proposed. The BDCWS algorithm maps a workflow constrained by user-defined deadline and budget values to cloud resources. BDCWS takes the basic characteristics of the cloud environment into accounts, such as heterogeneity, on-demand resource provisioning and pay-as-you-go price model with time periods as well as booting time, and take a series of targeted measures to improve its effectiveness. The experimental results show that compared with the latest two algorithms (DBWS and BDAS), our proposed BDCWS algorithm increases the chances of meeting deadlines and budget constraints. Especially when the deadline and budget are tight, the improvement of BDCWS algorithm is significantly higher than DBWS algorithm and BDAS algorithm. Besides, the BDCWS algorithm is more cost competitive than DBWS. In addition, we found that the structure of the workflow seems to have a significant impact on the success rate, such as the success rate of BDCWS algorithm in MONTAGE is higher than 90%, while on EPIGENOMICS it is only 81.7%. Therefore, in the future, we will study how to choose proper algorithm in an intelligent way for a given workflow to get better schedule. Another future direction is to explore the impact of VM performance variation on workflow execution and improve our strategy to adapt to volatile environments.

Acknowledgements This work is supported by National Natural Science Foundation of China (Grant Nos. 61772205, 61872084), Guangzhou Science and Technology Program key projects (Grant Nos. 202007040002, 201902010040 and 201907010001), Guangzhou Development Zone Science and Technology (Grant No. 2018GH17), Guangdong Major Project of Basic and Applied Basic Research (2019B030302002), and the Fundamental Research Funds for the Central Universities, SCUT (Grant No. 2019ZD26).

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interest regarding the publication of this paper.

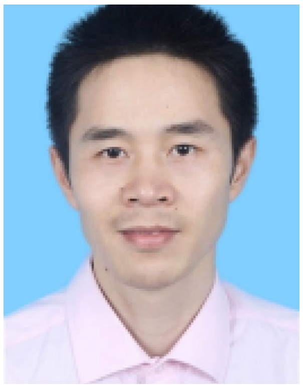
References

1. Chard, R., Chard, K., Bubendorfer, K., Lacinski, L., Madduri, R., Foster, I.: Cost-aware cloud provisioning. In: IEEE International Conference on E-Science 2015, pp. 136–144 (2015)
2. Lin, W., Xu, S., He, L., Li, J.: Multi-resource scheduling and power simulation for cloud computing. Inf. Sci. **397**(C), 168–186 (2017)
3. Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y., Wen, J.: Deadline-constrained cost optimization approaches for workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. **28**(12), 3401–3412 (2017)
4. Arabnejad, H., Barbosa, J.G.: A budget constrained scheduling algorithm for workflow applications. J. Grid Comput. **12**(4), 665–679 (2014)
5. Sakellariou, R., Zhao, H., Tsiakkouri, E., Dikaiakos, M.D.: Scheduling workflows with budget constraints. In: Integrated Research in GRID Computing. Springer, Boston, MA (2007)
6. Zheng, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control in market-oriented environments. In: International Workshop on Grid Economics and Business Models, pp. 105–119. Springer, Berlin (2011)
7. Zheng, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control. J. Grid Comput. **11**(4), 633–651 (2013)
8. Arabnejad, H., Barbosa, J.G., Prodan, R.: Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. Fut. Gener. Comput. Syst. **55**, 29–40 (2016)
9. Prodan, R., Wieczorek, M.: Bi-criteria scheduling of scientific grid workflows. IEEE Trans. Autom. Sci. Eng. **7**(2), 364–376 (2010)
10. Yu, J., Buyya, R., Tham, C.K.: QoS-based scheduling of workflow applications on service grids. In: Proc. of 1st IEEE International Conference on e-Science and Grid Computing 2005, pp. 5–8. IEEE CS Los Alamitos, CA (2005)
11. Cancan, L., Weimin, Z., Zhigang, L.: Path balance based heuristics for cost optimization in workflow scheduling. J. Softw. **24**(6), 1207–1221 (2013)
12. Chen, W., Xie, G., Li, R., Bai, Y., Fan, C., Li, K.: Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems. Fut. Gener. Comput. Syst. **74**(2017), 1–11 (2017)

13. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
14. Arabnejad, V., Bubendorfer, K., Ng, B.: Deadline distribution strategies for scientific workflow scheduling in commercial clouds. In: IEEE ACM International Conference Utility and Cloud Computing 2016, pp. 70–78 (2016)
15. Sahni, J., Vidyarthi, P.: A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. IEEE Trans. Cloud Comput. **6**(1), 2–18 (2018)
16. Ghafouri, R., Movaghar, A., Mohsenzadeh, M.: A budget constrained scheduling algorithm for executing workflow application in infrastructure as a service clouds. Peer-to-Peer Netw. Appl. **12**(1), 241–268 (2019)
17. Rodriguez, M.A., Buyya, R.: Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods. Acm Trans. Auton. Adapt. Syst. **12**(2), 1–22 (2017)
18. Shen, H., Li, X.: Algorithm for the cloud service workflow schedulingwith setup time and deadline constraints. J. Commun. **36**, 183–192 (2015)
19. Singh, V., Gupta, I., Jana, P.K.: A novel cost-efficient approach for deadline-constrained workflow scheduling by dynamic provisioning of resources. Fut. Gener. Comput. Syst. **79**(2018), 95–110 (2018)
20. Arabnejad, V., Bubendorfer, K., Ng, B.: Budget and deadline aware e-science workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. **30**(1), 29–44 (2019)
21. Ghasemzadeh, M., Arabnejad, H., Barbosa, J.G.: Deadline-budget constrained scheduling algorithm for scientific workflows in a cloud environment. In: international conference on principles of distributed systems 2017, pp. 1–16
22. Wu, F., Wu, Q., Tan, Y., Li, R., Wang, W.: PCP-B 2: partial critical path budget balanced scheduling algorithms for scientific workflow applications. Fut. Gener. Comput. Syst. **60**(2016), 22–34 (2016)
23. Sun, T., Xiao, C., Xu, X.: A scheduling algorithm using subdeadline for workflow applications under budget and deadline constrained. Cluster Comput. **22**(3), 5987–5996 (2019)
24. Wu, F., Wu, Q., Tan, Y.: Workflow scheduling in cloud: a survey. J. Supercomput. **71**(9), 3373–3418 (2015)
25. Alkhanak, E.N., Lee, S.P., Khan, S.U.R.: Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities. Fut. Gener. Comput. Syst. **50**(2015), 3–21 (2015)
26. Smanchat, S., Viriyapant, K.: Taxonomies of workflow scheduling problem and techniques in the cloud. Fut. Gener. Comput. Syst. **52**(2015), 1–12 (2015)
27. Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. J. Grid Comput. **14**(2), 217–264 (2016)
28. Rodriguez, M.A., Buyya, R.: A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments: workflow scheduling algorithms for clouds. Concurr. Comput. Pract. Exp. **29**(8), e4041 (2016)
29. Kaur, S., Bagga, P., Hans, R., Kaur, H.: Quality of Service (QoS) Aware Workflow Scheduling (WFS) in cloud computing: a systematic review. Arab. J. Sci. Eng **44**(4), 2867–2897 (2019)
30. Ming, M., Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: High Performance Computing, Networking, Storage & Analysis 2011, pp. 1–12
31. Abrishami, S., Naghibzadeh, M., Epema, D.H.J.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Fut. Gener. Comput. Syst. **29**(1), 158–169 (2013)
32. Abrishami, S., Naghibzadeh, M., Epema, D.H.J.: Cost-driven scheduling of grid workflows using partial critical paths. IEEE Trans. Parallel Distrib. Syst. **23**(8), 1400–1414 (2012)
33. Calheiros, R.N., Buyya, R.: Meeting deadlines of scientific workflows in public clouds with tasks replication. IEEE Trans. Parallel Distrib. Syst. **25**(7), 1787–1796 (2014)
34. Anwar, N., Deng, H.: Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments. Fut. Internet **10**(1), 5 (2018)
35. Meena, J., Kumar, M., Vardham, M.: Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint. IEEE Access **4**, 5065–5082 (2016)
36. Wu, C.Q., Lin, X., Yu, D., Xu, W., Li, L.: End-to-end delay minimization for scientific workflows in clouds under budget constraint. IEEE Trans. Cloud Comput. **3**(2), 169–181 (2015)
37. Arabnejad, V., Bubendorfer, K., Ng, B.: Budget distribution strategies for scientific workflow scheduling in commercial clouds. In: International Conference on E-science 2016, pp. 137–146
38. Faragardi, H.R., Sedghpour, M.R.S., Fazliahmadi, S., Fahringer, T., Rasouli, N.: GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds. IEEE Trans. Parallel Distrib. Syst. **31**(6), 1239–1254 (2019)
39. Rizvi, N., Ramesh, D.: Fair budget constrained workflow scheduling approach for heterogeneous clouds. Cluster Comput. 1–17 (2020).
40. Chakravarthi, K.K., Shyamala, L., Vaidehi, V.: Budget aware scheduling algorithm for workflow applications in IaaS clouds. Cluster Comput. 1–15 (2020).
41. Su, S., Jian, L., Huang, Q., Xiao, H., Kai, S., Jie, W.: Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput. **39**(4–5), 177–188 (2013)
42. Topcuoglu, H., Hariri, S., Wu, M.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
43. Zhu, Z., Zhang, G., Li, M., Liu, X.: Evolutionary multi-objective Workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1344–1357 (2016)
44. Choudhary, A., Gupta, I., Singh, V., Jana, P.K.: A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Fut. Gener. Comput. Syst. **83**, 14–26 (2018)
45. Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Fut. Gener. Comput. Syst. **48**, 1–18 (2015)
46. Verma, A., Kaushal, S.: Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. In: Engineering & Computational Sciences 2014, pp. 1–6
47. Verma, A., Kaushal, S.: Cost-time efficient scheduling plan for executing workflows in the cloud. J. Grid Comput. **13**(4), 1–12 (2015)
48. Amazon: Amazon EC2 Pricing. https://aws.amazon.com/ec2/pricing/. Accessed 5 Aug. 2019
49. Google: Google Cloud Platform. https://cloud.google.com/compute/ (2017). Accessed 5 Aug 2019
50. Microsoft: Microsoft Azure. https://azure.microsoft.com (2017). Accessed 5 Aug 2019
51. Barr, J.: New-Per-Second Billing for EC2 Instances and EBS Volumes. https://aws.amazon.com/tw/blogs/aws/new-per-second-billing-for-ec2-instances-and-ebs-volumes/ (2017). Accessed 1 Feb 2019
52. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Fut. Gener. Comput. Syst. **29**(3), 682–692 (2013)
53. Palankar, M.R., Iamnitchi, A., Ripeanu, M., Garfinkel, S.: Amazon S3 for science grids: a viable solution? In: Proceedings

of the 2008 International Workshop on Data-Aware Distributed Computing 2008, pp. 55–64. ACM
54. Mao, M., Humphrey, M.: A performance study on the VM startup time in the cloud. In: International Conference on Cloud Computing 2012, pp. 423–430
55. Juve, G.: Workflow Generator. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator. Accessed 12 June 2018

Naqin Zhou received the Ph.D. degree at South China University of Technology. Now she is a Lecturer of Cyberspace Institute of Advanced technology, Guangzhou University. Her research interests are mainly on parallel computing, cloud computing and grid computing systems.



Weiwei Lin received his B.S. and M.S. degrees from Nanchang University in 2001 and 2004, respectively, and the PhD degree in Computer Application from South China University of Technology in 2007. Currently, he is a professor in the School of Computer Science and Engineering at South China University of Technology. His research interests include distributed systems, cloud computing, big data computing and AI application technologies. He has published more than 100 papers in refereed journals and conference proceedings. He is a senior member of CCF.

Wei Feng is working at Guangzhou Branch of Shanghai Yizhong Enterprise Management Consulting Co., Ltd. He has long been engaging in the construction of government cloud computing platform for intelligent city. His research interests include distributed system, grid computing and cloud computing.

Fang Shi received the master degree in engineering from Guangzhou University in 2019. She is currently a Ph.D. student in the School of Computer Science and Engineering at South China University of Technology. Her research interests include cloud computing and big data

Xiongwen Pang received his Bachelor from ChongQing Univeristy in 1994, M.S. degrees from Harbin Institute of Technology in 1996, and the PhD degree in Computer Application from South China University of Technology in 2007. Currently, he is an assosiate professor in the School of Computer Science, South China Normal University. His research interests include big data, deep learning and AI application technologies. He has published more than 20 papers in refereed journals and conference proceedings. He is a senior member of CCF.