Cluster Computing
https://doi.org/10.1007/s10586-018-1751-9

A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained

Ting Sun[1] · Chuangbai Xiao[1] · Xiujie Xu[1,2]

Received: 31 October 2017 / Revised: 4 January 2018 / Accepted: 6 January 2018

Abstract
With the development of the cloud and grid computing, the cloud infrastructures and grids provide a platform for workflow applications. It is very essential to meet the requirements of users and to complete workflow scheduling efficiently. The scheduling of the workflow is limited by quality of service (QoS) parameters. Many scheduling algorithms have been proposed for the execution of workflow applications using QoS parameters. In this study, we improved a scheduling algorithm that considers workflow applications under budget and deadline constraints. This algorithm provided a simple way to deal with the deadline and budget constraints. The algorithm was named BDSD and used to find a scheduling that satisfies of deadline and budget constraints required by a user. The planning success rate (PSR) was utilized to show the effectiveness of the proposed algorithm. For the simulation experiment, random and real workflow applications were exploited. Experimental results showed that compared with other algorithms the algorithm had a higher PSR.

Keywords Scheduling · Sub-deadline · Quality of service · Planning success rate · Workflow application

1 Introduction

Studies on the workflow scheduling problem have focused on heterogeneous computing systems and distributed environments, such as grids and clusters. The challenges in the workflow scheduling problem are the diverse quality of service (QoS) requirements. The workflow can be defined by directional acyclic graph (DAG), which includes a large number of priority constraints and parallel tasks. Effective scheduling is very important in executing applications on heterogeneous environments. Workflow scheduling that satisfies QoS parameter has become important in practice computing. Yu et al. [1] presented several classical workflow scheduling algorithms. The algorithms almost consider a single objective, such as minimising cost or makespan. Numerous algorithms consider multiple QoS parameters due to the increasing QoS reguirements of users. Several scheduling algorithms adopt methods that consists two QoS parameters at the same time, for example consider time and cost together; consequently, the problem becomes increasingly challenging [2]. It has been concluded that the DAG scheduling problem is an NP-hard problem [3]. Scheduling tasks for any number of processors and scheduling the weights of one or two units to two processors are proven to be NP-hard problems [4].

The two primary optimisation problems of minimising cost under deadline constraints and minimising mak-espan under budget constraints have also been studied. Scheduling problems are of different types, such as best-effort, deadline-constrained, budget-constrained, robust, multi-criteria, data-intensive, energy-aware and hybrid-resource workflow scheduling [5]. In a cloud environment, a cloud is distributed computer cluster, it through the network to the remote user to provide on-demand computing resources and services [6]. A cloud user can make many cloud service at the same time. The cloud has the autonomic feature, and the VMs have the diversity feature. In infrastructure as a service (IaaS) cloud computing, computing resources are provided to remote users in the form of leases. Li and Qiu et al. [7] considered IaaS cloud system and proposed two kinds of online

This paper has been presented in the 4th IEEE International Conference on Cyber Security and Cloud Computing (IEEE cscloud 2017).

✉ Ting Sun
sunting07@emails.bjut.edu.cn

1 School of Computer Science, Beijing University of Technology, Beijing 100124, People's Republic of China

2 School of Management Engineering, Shandong Jianzhu University, Jinan 250101, People's Republic of China

dynamic resource allocation algorithm. They adjust the allocation of resources by obtaining the latest information on the execution of the task, which is a dynamic approach. Qiu et al. [8] presented a genetics-based optimisation algorithm for chip multiprocessors in green clouds; the algorithm uses phase-change memory technique. Gai, Qiu et al. [9] used a genetic algorithm to develop a multimedia data distribution of heterogeneous memory in cloud computing based on cost-aware. Xu et al. [10] adopted the slot extension backfill strategy to make full use of the time slot amongst tasks but did not consider the problem of scheduling costs.

Many studies have been conducted on multi-objective scheduling [11–20]. If there is no suitable processor that satisfies the first constraint of the task being executed, then it may be select the processor according to the second constraint of the task. In [16], the tasks in the DAG are divided into different levels according to the different sub-deadline of the task, and then the workflow is scheduled. If the tasks had the same sub-deadline, then they were in the same level. In [17], the workflow was grouped into sub workflows, and deadline was divided into different partitions. In [11], the sub-deadline value was used with the time quality factor in the processor selection phase.

Arabnejad et al. [11] proposed a schedule workflow application (DBCS) for computational heterogeneity constrained to two QoS parameters. They determined a schedule that satisfied the deadline and cost constraint by the user. The algorithm was the first to implement low-time complexity scheduling with two QoS parameters. Zhang and Sakellariou [12] proposed a scheduling algorithm, which is optimized for multi-heterogeneous environment with time constraints of the workflow scheduling. Arabnejad and Barbosa [13] also proposed an algorithm (HBCS) that minimizes the scheduling time of the task and allows the user to execute the cost at a given cost. This algorithm allows numerous processors to be considered and selected. Unlike other algorithms, this algorithm does not use the earliest completion time strategy to select the processor. For processor selection, the algorithm uses a function that includes time and cost in order to select the processor for the current task.

In this study, a DAG scheduling model in linear programming is built. We improve a scheduling algorithm with two QoS parameters and obtain a performance that is better than that of other algorithms. The sub-deadline is used to define task priority, which is the difference of our algorithm from others. Then, we obtain results for real-world applications and randomly generate a DAG. The results show that BDSD algorithm has a low time complexity and obtains a higher planning success rate than other algorithms, namely, DBCS, BHEFT and HBCS.

The remainder of this paper is organized as follows: Section 2 is devoted to related work. Section 3, presents the workflow management system and the system model. Section 4 describes the BDSD algorithm and gives an example to describe it. Section 5, provides the results of simulation experiments to illustrate the advantages of the BDSD algorithm. Our results and future research work are described in the final section.

2 Related work

For the workflow scheduling problem, it can be divided into single target, QoS constraint and multi-objective workflow scheduling according to the scheduling objective. QoS-constrained workflow scheduling approximates actual applications. This problem is generally based on the constraints of several objectives to optimise other objectives. In all workflow scheduling problems, workflow scheduling with deadline constraints and workflow scheduling with budget constraints are the two most studied scheduling problems.

Workflow scheduling based on deadline constraints implies that the planner sets the deadline as a constraint. Most users generally consider the cost in a cloud environment. The performances and prices of cloud resources are different. Users wish to spend the minimum cost but do not want more than a given period. To solve this problem, the planner is usually under the constraints of cost minimisation. The general form is as follows:

$$\text{Min} \quad \text{C} \tag{1}$$

$$\text{s.t.} \quad \text{makespan} \leqslant \text{D} \tag{2}$$

The scheduling strategy of the scheduling problem can be divided into meta-heuristic and deadline distribution-based heuristic.

For meta-heuristic methods of workflow scheduling, Yu and Buyya [21] proposed a genetic algorithm. They also put forward a fitness function that include cost and time factor. The function can reduce the cost as much as possible and does not violate deadline constraints. Wu et al. [22] presented a workflow scheduling algorithm based on revised discrete particle swarm optimisation (PSO). Rodrigues [23] solved the problem of workflow scheduling on the IaaS cloud by using the meta-heuristic optimisation algorithm of PSO.

Deadline distribution-based heuristic is divided into three stages: task sorting, deadline allocation and resource selection. Compared with the single-objective workflow scheduling algorithm, this strategy increases the deadline allocation phase. DTL [16], DBL [24] and DET [25] are heuristic algorithms for the deadline allocation strategy. The DTL algorithm minimises the user cost and meets the user deadline at the same time. This algorithm also rearranges the unallocated task to adopt delay. DTL assigns tasks as synchronisation or simple tasks. In the process of resource allocation, the DTL algorithm uses the Markov decision process to optimise the task scheduling problem. The DBL algorithm adopts the bottom-up approach to reverse a task and allocates the time difference to each layer to increase the cost optimisation interval of the task. The DET algorithm divides a task into different paths. In the deadline allocation phase, the deadline is divided into the mission task, and the iterative algorithm plays a key role in finding time windows for non-mission tasks without changing the priority constraints between the tasks. In the processor selection process, DET uses dynamic method to find resources.

Another problem is that users wish to complete the workflow scheduling as soon as possible. Therefore, the workflow scheduling completion time becomes an important goal. Costs become a secondary optimisation goal, as long as the cost cannot exceed the given budget. This problem generally minimises the completion time (makespan) under a given budget, and the general form is as follows:

$$\text{Min} \quad \text{makespan} \tag{3}$$

$$\text{s.t.} \quad C \leqslant B \tag{4}$$

The scheduling strategy of this problem can be divided into one-time heuristic and back-tracking heuristic.

DBCS [11], BHEFT [12], HBCS [13] and GreedyTimeCD [15] are one-time heuristic algorithms. These algorithms use the idea of HEFT algorithm, and the time complexity of these algorithms has not changed compared with HEFT algorithm. The budget of the BHEFT algorithm is dynamically distributed. The algorithm proposes the current task budget CTB and the remaining budget SAB and allocates the current remaining budget to the unallocated task according to the AET ratio of the unallocated task. The HBCS algorithm selects the cost and execution time with the largest weight of resources. DBCS selects the aggregation weight in the time factor added sub-term constraints and the largest weight of the resources. The GreedyTimeCD algorithm statically assigns all tasks with the user-given budget before scheduling and then selects resources.

Multi-objective workflow scheduling generally considers multiple goals, and all these goals are important. Many people transform multiple objectives into a single objective to solve the problem. Scheduling methods include aggregation and Pareto methods.

3 Problem description

3.1 Workflow management system

Users provide the QoS requirements for workflow application and pay the service costs for the workflow tasks. Moreover, resource owners provide the services and obtain the cost from the users. The planners receive the workflow and associated constraints. They find the right resources for all tasks of the workflow and thus create a workflow plan. Workflow planning is based on the user's constraints for each task to select a resource. To submit tasks to resources, the planner needs to determine the resources and time needed to run the task on the resources. Therefore, the planner must study the information of users and resource owners and make a reasonable workflow plan. The user is satisfied with the arrangement under the constraints, and the resource owner obtains as much revenue as possible.

The planner determines whether the user's requirements are met based on the parameters of the existing resource. These parameters include the performance and quantity of resources. If the constraints are not satisfied, the planner can reject the workflow; otherwise, the planner accepts the workflow. The resource owner hopes to perform every new workflow and access and execute the resources. In this study, the resource owner provides resources and wants all resources to be maximized, which means to acquire as much revenue as possible. The user gives the workflow with the budget C and deadline D constraints and expects to schedule all workflow tasks to the resources. The planner provides the scheduling algorithm to the user and resource owner.

3.2 Problem definition and system model

Use a directed figure with no cycles (DAG) to represent a workflow. The DAG can be modelled by G = (V, E), where V is the set containing n tasks and E represent the set of edges containing m edges. Edge (i, j) ∈ E is directional, indicating the order in which the task is executed which means task n_i is executed first than task n_j starts. A is a n × n communication matrix, where A_{ij} is the data from task n_i to task n_j. Let t_u is the time to transmit a data unit, the communication time between tasks n_i and task n_j is $\bar{A}_{ij} = A_{ij} \times t_u$.

In the DAG, the entry task is the task with no a parent node, the exit task is a task with no a child node. For most of the task scheduling algorithms, there is only one input and an output of the task map. So in this article we also assume that DAG only has an entry task and an exit task. If there are multiple entry or exit tasks, we add to the graph with zero weight and zero communication edges. It also ensures that the figure has a single entry and single exit task.

Let P = $\{p_1, p_2 \cdots p_m\}$ has m processors. In this model, each resource can serve any task of DAG with every type of service. W is a n × m computation cost matrix, and w_{ij} is the execution time of task n_i on processor p_j. Each processor has its own price under unit execution time R = $\{r_1, r_2 \cdots r_m\}$, so the cost of task n_i on processor p_j is $c_{ij} = w_{ij} \times r_j$. We use makespan to represent the total schedule length of the DAG, it can denote the completion time of the exit task in the

DAG. Define the workflow makespan by [14]: makespan $= max\left\{AFT_{n_{exit}}\right\}$, where $AFT_{n_{exit}}$ denotes the actual finish time of the exit task n_{exit}.

Assume that C(budget cost) and D(deadline) are given, we wish to decide if there exists a schedule of total cost at most C such that the completion time of exit task assigned to processor is at most D. We denote

$$x_{ij} = \begin{cases} 1, & if\ task\ n_i\ is\ arranged\ on\ processor\ p_j \\ 0, & \text{else} \end{cases}$$

ST and *AFT* denote the start time and the completion time of task n_i on the processor assigned by the scheduling algorithm. For every task n_i, we use its sub-deadline to sort the tasks. This scheduling problem is descried as follows:

$$\text{(P)} \quad \sum_{j=1}^{m} x_{ij} = 1, i = 1, 2, \ldots n, \tag{5}$$

$$\max(\mathrm{AFT}_{n_{exit}}) \leq \mathrm{d},\ j = 1, 2, \ldots m, \tag{6}$$

$$\sum_{j=1}^{m} \sum_{i=1}^{n} c_{ij}\mathrm{x}_{ij} \leq \mathrm{C}, \tag{7}$$

$$(\mathrm{ST_{ij}} + w_{ij})\mathrm{x}_{ij} \leq \mathrm{d}, i = 1, 2, \ldots n,\ j = 1, 2, \ldots m, \tag{8}$$

$$\mathrm{x}_{ij} \in \{0, 1\} \tag{9}$$

where (5) means every task n_i must be assigned on the processor; (6) means the completion time of the exit task no more than d, where d $\leq$ D; (7) means the total cost of all tasks on processors at most C; (8) means the task n_i completion time on the processor no more than d.

4 Proposed budget-deadline constrained based on sub-deadline scheduling algorithm

In this section, to find a viable workflow schedule within the user's given cost and time limit, the budget deadline constraints are presented based on the sub-deadline(*SDL*) scheduling algorithm. By using the known HEFT algorithm, the scheduling method is divided into the sort of user tasks and choice of processor. Based on heuristics of the algorithm, the BDSD algorithm also consists of two parts: the sort of tasks and the choice of the processor. The algorithm determines the priority of the task based on the *SDL* of the task in the DAG. The algorithm aims to make the schedule successful under the budget and deadline constraints. If the time and budget constraints are met, it means that the schedule is successful; otherwise, the schedule is a failure. The BDSD is shown in Algorithm 1. Before the description of the algorithm, there are some definitions.

Definition 1 [14] Define the Earliest Start Time (*EST*) of a task n_i on processor p_j as

$$EST\left(n_i, p_j\right) = \max\left\{T_{avail[j]}, max_{n_k \in pred(n_i)}\left(\mathrm{AFT}\left(n_k\right) + \bar{A}_{k,i}\right)\right\} \tag{10}$$

Denote the Earliest Finish Time(*EFT*) of a task n_i on processor p_j by

$$EFT\left(n_i, p_j\right) = EST\left(n_i, p_j\right) + \omega_{i,j} \tag{11}$$

where $pred\ (n_i)$ represents all predecessor task set of task n_i. $T_{avail[j]}$ is the earliest time for processor p_j to complete all previous tasks to prepare for the next task. For task n_{entry}, $EST\left(n_{entry}, p_j\right) = 0$. If task n_k and task n_i are assigned on the same processor, the communication time $\bar{A}_{k,i}$ is 0. In Eq. (10), the inner max part is the time for all data required by n_i to reach processor p_j.

4.1 Task sorting

In the HEFT algorithm, the tasks are sorted according to their priorities by computing the upward rank ($rank_u$) [14]. In the BDSD algorithm, to sort all tasks *SDL* was calculated and defined as follows:

$$SDL\ (n_i) = min_{n_k \in succ(n_i)}\left[SDL\ (n_k) - \bar{A}_{i,k} - ET_{min}\ (n_k)\right] \tag{12}$$

where $succ\ (n_i)$ represents the set of all successor tasks of task n_i and $ET_{min}\ (n_k)$ is the minimum execution time for all processor scheduling task n_k. For task n_{exit}, $SDL\ (n_{exit}) =$ D.

In this study, the *SDL* of each task was calculated. The values of *SDL* are sorted in ascending order, and the corresponding task obtains a new sequence.

4.2 Processor selection

To consider how to select the processor for each task, time and cost related variables were considered.

Definition 2 Define the user remaining budget by*rb* which means that the budget can be used for the remaining unscheduled tasks:

$$rb = rb - c_{ij}, \tag{13}$$

where the initial value $rb =$ C.

Denote l as the remaining unscheduled task numbers. Define the expect reasonable cost(erc) for task n_i:

$$erc_{n_i} = \frac{rb}{l}. \tag{14}$$

According to the above variables, some processors that cost high can be deleted. We can only consider the processors whose cost satisfies $c_{k,p} \leq erc_k$:

$$\omega_k = \left\{ w_{x,p} | \exists w_{x,p}, c_{k,p} \leq erc_k \right\}. \tag{15}$$

Then, the selection rules can be used to select the best possible processor. The rules are as follows:

We select the processor with the earliest finish time.

1. If $\omega_k = \emptyset$, we select the processor with the earliest finish time;
2. If $\omega_k = \emptyset$, we select the processor with the cheapest cost.

4.3 Detailed description of the BDSD algorithm

Algorithm 1: BDSD scheduling algorithm

```
Input DAG G, the value given by the user for the budget C and
  the deadline D;
1: Calculate all tasks $SDL$ according to Eq. (12)
2: Enter the initial values for $rb$ and $l$, $rb = C$, $l =$ n
3: while there is task that can be scheduled do
4: $n_i =$ the first task is sorted according to the$SDL$ value
5: Calculate the $erc$ for task $n_i$ according to Eq. (14)
6: According to steps 5 and 6, the set $\omega_k$ for task $n_i$ is
  given by Eq. (15)
7: for the task $n_i$ can choose the processor
8: Calculate the earliest completion time of task $n_i$
  which processor in the $\omega_k$
9: end for
10: Choice a processor for task $n_i$ based on the selection rules
11: Update remaining budget $rb$ according to Eq. (13),
  update the $l$, $l = l - 1$
12: end while
Return schedule map.
```

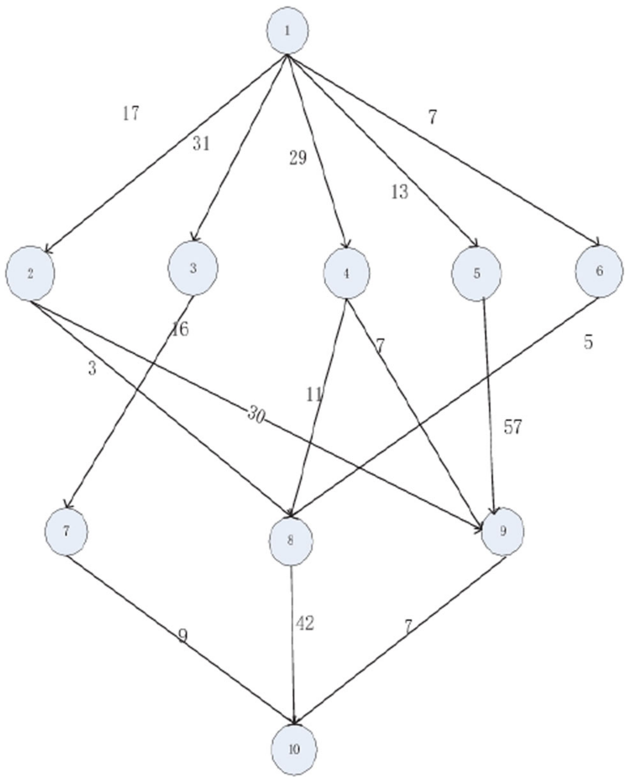

Fig. 1 An example DAG with 10 tasks

Table 1 Computation time of tasks on processors

n_i/P_j	P_1	P_2	P_3
n_1	22	21	36
n_2	22	18	18
n_3	32	27	43
n_4	7	10	4
n_5	29	27	35
n_6	26	17	24
n_7	14	25	30
n_8	29	23	36
n_9	15	21	8
n_{10}	13	16	33

4.4 An example

An illustrative example is provided to better understand the BDSD algorithm. Figure 1 shows the DAG. The figure shows the structure of the DAG with 10 task nodes and two tasks in different processors on the data transfer time. If there are two tasks on a processor, we assume that their data transfer is 0. For simplicity, let $t_u = 1$, so the communication time between tasks n_i and task n_j is $\bar{A}_{ij} = A_{ij} \times 1$. The task computation time in each processor is shown in Table 1. Each processor's own price under unit execution time is R $= \{0.92, 0.29, 0.40\}$. The computation cost of task n_i on processor p_j is shown in Table 2. We assume that the user's deadline is 200 and the budget is 95, which means that the user wants to complete the DAG schedule under this condition.

First, the SDL of the DAG tasks is calculated as defined in Eq. (12). Table 3 shows the values of the SDL and the sequence of tasks. The new task sequence is obtained in ascending order of the SDL values. From Table 3, the new sequence is as follows: L $= \{n_1, n_4, n_5, n_6, n_2, n_8, n_3, n_7, n_9, n_{10}\}$. By taking the highest priority task in scheduling sequence L, the erc can be calculated for each task as defined in Eq. (14). Additionally, the earliest finish time of the current task is calculated as defined in Eq. (11).

Table 2 Computation costs of tasks on processors

n_i/P_j	P_1	P_2	P_3
n_1	20.24	6.09	14.40
n_2	20.24	5.22	7.20
n_3	29.44	7.83	17.20
n_4	6.44	2.90	1.60
n_5	26.68	7.83	14.00
n_6	23.92	4.93	9.60
n_7	12.88	7.25	12.00
n_8	26.68	6.67	14.40
n_9	13.80	6.09	3.20
n_{10}	11.96	4.64	13.20

We assume that if two tasks are executed on the same processor, then the computation time of transfer between the two tasks are 0.

Task n_1 starts the first schedule. $erc\,(n_1) = \frac{95}{10} = 9.5$. Only processor p_2 satisfies the condition $c_{12} = 6.09 < 9.5$. Therefore, for task n_1 we choose processor p_2. The *EST* of task n_1 on the processor is 0, and the *EFT* of task n_1 on processor p_2 is $EFT\,(n_1, p_2) = 0 + 21 = 21$.

Task n_4 has the second priority, so it starts the second schedule. $erc\,(n_4) = \frac{95-6.09}{9} = 9.87$. The processors p_1, p_2, p_3 satisfy the condition $c_{4j} < 9.87$. The *EST* of task n_4 on processors p_1, p_2, p_3 is 0, 21 and 0, respectively. Then, we calculate the *EFT* of task n_4 and find the minimum *EFT*, which is 31, and choose the processor p_2.

Task n_5 starts the third schedule.$erc\,(n_5) = \frac{95-6.09-2.9}{8} = 10.75$. Processor p_2 satisfies the condition $c_{52} < 10.75$. *EST* of task n_5 on processor p_2 is 31. Then, *EFT* of task n_5 is calculated, and the minimum *EFT* is determined, which is 58, so processor p_2 is chosen.

Task n_6 starts the fourth schedule. $erc\,(n_6) = \frac{95-6.09-2.9-7.83}{7} = 11.17$. Processors p_2 and p_3 satisfy the condition $c_{6j} < 11.17$. *EST* of task n_6 on processor p_3 is 28. Then *EFT* of task n_6 is calculated, and the minimum *EFT* is determined, which is 52, so processor p_3 is chosen.

The remaining tasks in the newlist are calculated and shown in Table 4. Table 4 shows the results of BDSD on the example of DAG in Fig. 1. In the Table 4, each task selects the processor, and the task cost on the selected processor is also shown. As shown in Table 4, the total cost of completing the schedule is 63.21, which is less than 95. The makespan of this scheduling is 164, which is less than 200, so the condition is satisfied.

Table 4 An example to illustrate the steps of BDSD using the workflow in Fig. 1

task	erc	ω_i	select	EST	EFT	cost
n_1	9.50	$\omega_{1,2}$	$\omega_{1,2}$	0	21	6.09
n_4	9.87	$\omega_{4,1}, \omega_{4,2}\, \omega_{4,3}$	$\omega_{4,2}$	21	31	2.90
n_5	10.75	$\omega_{5,2},$	$\omega_{5,2}$	31	58	7.83
n_6	11.17	$\omega_{6,2}, \omega_{6,3}$	$\omega_{6,3}$	28	52	9.60
n_2	11.43	$\omega_{2,2}, \omega_{2,3}$	$\omega_{2,3}$	52	70	7.20
n_8	12.28	$\omega_{8,2}$	$\omega_{8,2}$	73	96	6.67
n_3	13.68	$\omega_{3,2}$	$\omega_{3,2}$	96	123	7.83
n_7	15.63	$\omega_{7,1}, \omega_{7,2}\, \omega_{7,3}$	$\omega_{7,2}$	123	148	7.25
n_9	19.82	$\omega_{9,1}, \omega_{9,2}\, \omega_{9,3}$	$\omega_{9,3}$	115	123	3.20
n_{10}	36.43	$\omega_{10,1}, \omega_{10,2}\, \omega_{10,3}$	$\omega_{10,2}$	148	164	4.64
						63.21

5 Experimental results and discussion

In this section, we present the PSR comparisons of the BDSD algorithm with the DBCS [11], BHEFT [12] and HBCS [13] algorithms. For this purpose, we consider two types of DAGs for the experiment: one is randomly generated by the DAG generator workflow, and the other is the actual workflow, which represents some practical problems. SimGrid [18] provides a DAG generator, which can use some parameters to simulate the actual problem into DAG.

5.1 Workflow structure

The DAG can be randomly generated and the generator program can acquire from :https://github.com/frs69wq/daggen. It is required three forms to model a task of computational complexity, which represent many common application: $a.d$, $a.d \backslash \log d$ and $d^{3/2}$, where a is chosen randomly between 2^6 and 2^9.

The DAG shape requires a random DAG generator to define some parameters. n is used to represent the number of tasks in the DAG, and *fat* to indicate the height and width of DAG. The width of the DAG is also the number of tasks that can be executed at the same time. A small width means

Table 3 The sub-deadline of the DAG tasks and the task newlist

task	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}
SDL	75	119	148	111	115	117	178	145	180	200
increase-SDL	75	111	115	117	119	145	148	178	180	200
task- newlist	n_1	n_4	n_5	n_6	n_2	n_8	n_3	n_7	n_9	n_{10}

that the DAG has a low task parallelism, while a large width means that the DAG has a high task parallelism. The density is used to indicate the number of edges between the two levels of the DAG. We know that the density value is low, indicating that the edge is small, and a large value means that the DAG has many edges. Regularity is used to represent the consistency of the number of nodes in each level. If the regularity is small, it means that a level contains different tasks. On the contrary, if the regularity is large, it means that all levels involve the same number of tasks. J means that an edge can be selected from m to $m + J$.

In our experiment, we considered the following parameters for random DAG:

n = [10, 20, 30, 50, 100];
J = [1, 3];
regularity = [0.2, 0.3];
fat = [0.1, 0.3];
density = [0.3, 0.8];
CCR = [0.5, 1];

We randomly selected these parameters to generate different DAGs. We also took into account the real situation of DAGs, such as Montage, Sipht, CyberShake and Epigenomics.

To understand the performance of each scheduling algorithm, the experiment was repeated many times, and PSR was used to describe the result. If the time and cost constraints satisfy the conditions, the schedule is successful. If they are not met, the schedule is a failure. PSR is defined as follows:

$$PSR = \frac{the\,times\,of\,successful\,schedule}{number\,of\,repeated\,experiment} \times 100$$

For a DAG, we provide rational values for deadline and budget constraints following the principles. In general, assume that the task starts at time 0. By using the HEFT algorithm [14], the DAG can obtain makespan M. The deadline constraint D $=$ M $+ f_d \times (2M)$, where $0 \le f_d \le 1$. By considering budget constraint B, we define it as follows: B $=$ L $+ f_b \times (U - L)$, where L is the total cost of each task assigned to the lowest cost processor, U is the total cost of each task assigned to the highest cost processor and $0 \le f_b \le 1$.

5.2 Experimental results

In the experiments, we consider scheduling under different types of constraints. The results for real-world DAGs and randomly generated DAGs are shown. In the environment using three heterogeneous resources for simulation, each one has a processing capability (a unit of calculation for each unit of time) randomly acquired from the interval (10, 100). The

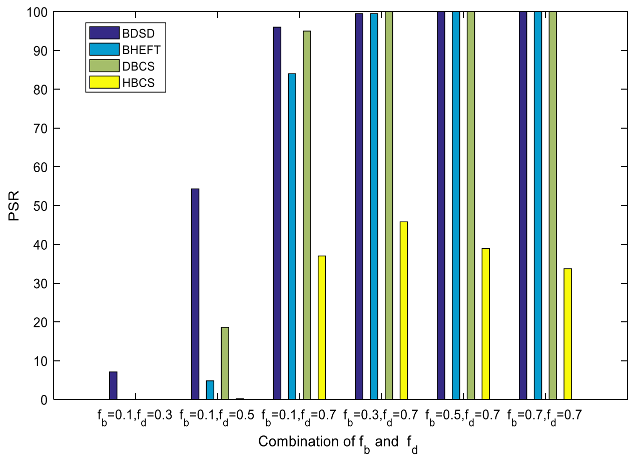

Fig. 2 Sipht with 30 nodes, PSR with different constraints

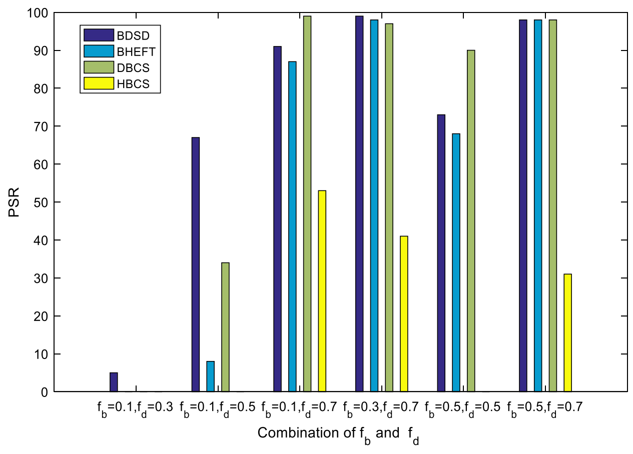

Fig. 3 Inspiral with 30 nodes, PSR with different constraints

bandwidth of each link is randomly selected from the interval (10, 100), and the resources are connected by heterogeneous networks. The computation cost of each task is randomly selected from the interval (1, 100). The definition of CCR is the ratio of the amount of communication between two tasks and the amount of computation executed in the DAG. We use CCR of 0.5 and 1 for simulation.

Use five DAGs that correspond to the real-world workflow application:

sipht: 30 tasks;
Inspiral: 30 tasks;
Montage: 50 tasks;
CyberShake: 50 tasks;
Epigenomics: 100 tasks;

Figures 2, 3, 4, 5 and 6 show the results of the real-world DAGs and obtained PSR values. We consider DAG scheduling under the constraints of different parameters.

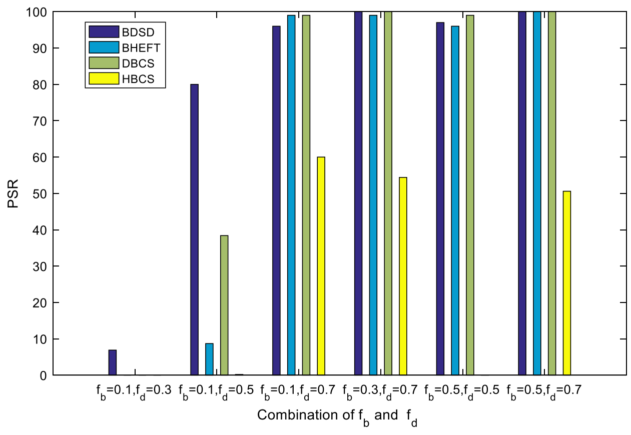

Fig. 4 Montage with 50 nodes, PSR with different constraints

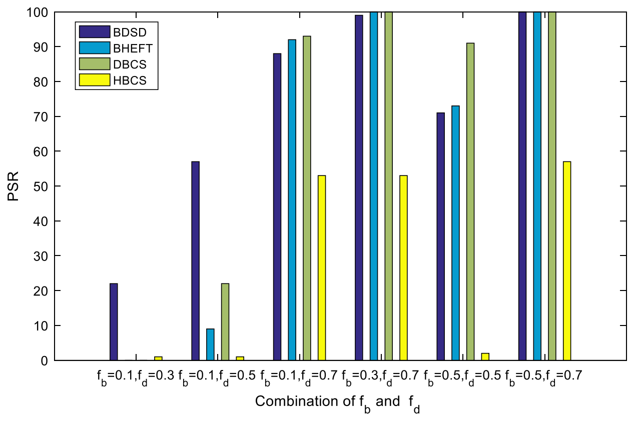

Fig. 5 CyberShake with 50 nodes, PSR with different constraints

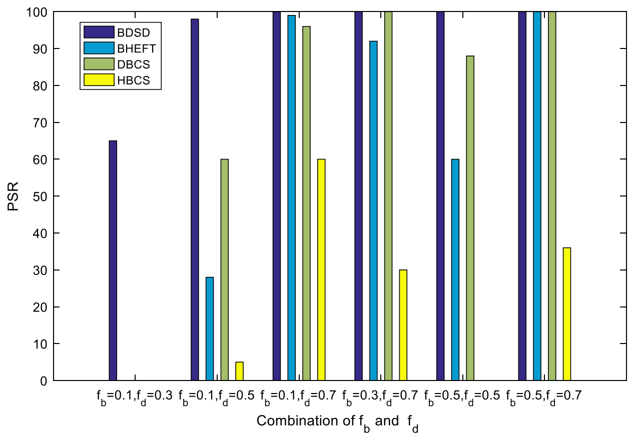

Fig. 6 Epigenomics with 100 nodes, PSR with different constraints

The results for Sipht DAG are shown in Fig. 2. We provide different combinations of cost factor f_b and deadline factor f_d. As shown in Fig. 2, when f_b and f_d are relatively small, (i.e. the cost and time are relatively tight), PSR of the BDSD algorithm is relatively high. When f_b and f_d are relatively large, (i.e. the cost and time are relatively loose), PSR of the BDSD algorithm is consistent with the BHEFT and DSCS

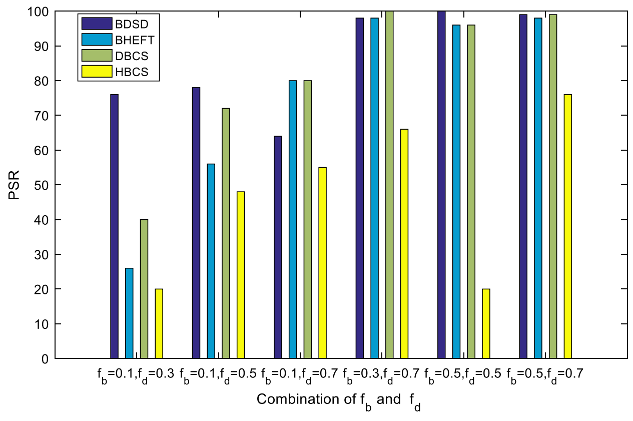

Fig. 7 Randomly generated DAG with 10 nodes, PSR with different constraints

algorithms. Compared with that in other algorithms, PSR in the HBCS algorithm is lower. The results for the Inspiral DAG are shown in Fig. 3. The figure shows that the BDSD algorithm is better than the other algorithms when f_b and f_d are relatively small. The results for Montage DAG with 50 nodes are shown in Fig. 4. Figure 5 shows the CyberShake with 50 nodes, which obtained the PSR. Figure 6 shows the results of the Epigenomics DAG. The figures show that the BDSD algorithm achieved the best performance compared with the other three algorithms, especially when $f_b = 0.1$ and $f_d = 0.5$ (i.e. the budget and deadline constraints are tight). When $f_b = 0.5$, $f_d = 0.5$, $f_b = 0.5$ and $f_d = 0.7$, the PSR values of the BDSD, BHEFT and DBCS algorithms are higher than that of the HBCS algorithm.

Figures 7, 8, 9, 10 and 11 show the results of the randomly generated DAGs and obtained PSR values. We also consider six combinations of different types of constraints and five randomly generated DAGs with 10, 20, 30, 50 and 100 tasks, where CCR $= 0.5$. As shown in Fig. 7, when the budget parameter f_b is equal to 0.1 and the deadline parameter f_d is equal to 0.3, the BDSD algorithm showed the best performance compared with the other algorithms. However, when the values of f_b and f_d increase, BDSD showed the same level of performance with the other algorithms. The PSR value of HBCS is lower than that of the other algorithms. In Figs. 8, 9, 10, and 11, as the number of DAG nodes increases, the experimental results are the same as those seen in Fig. 7. Therefore, in the randomly generated DAG experiments, the BDSD algorithm showed the best performance among the other algorithms when both budget and deadline constraints are tight.

Based on all the figures, the BDSD algorithm is better than the other three algorithms, especially when budget and deadline constraints are relatively small. With increasing f_b and f_d, the PSR values of the BDSD, BHEFT and DBCS algorithms become large. However, in the HBCS algorithm, the PSR value is still low.

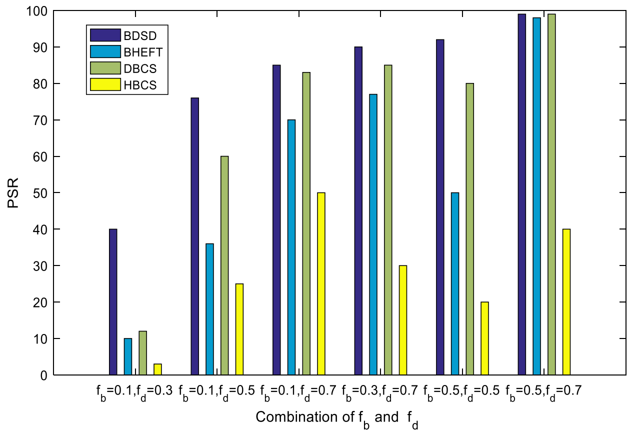

Fig. 8 Randomly generated DAG with 20 nodes, PSR with different constraints

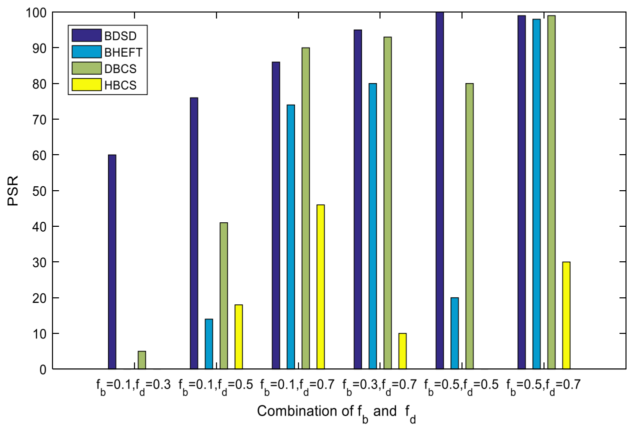

Fig. 9 Randomly generated DAG with 30 nodes, PSR with different constraints

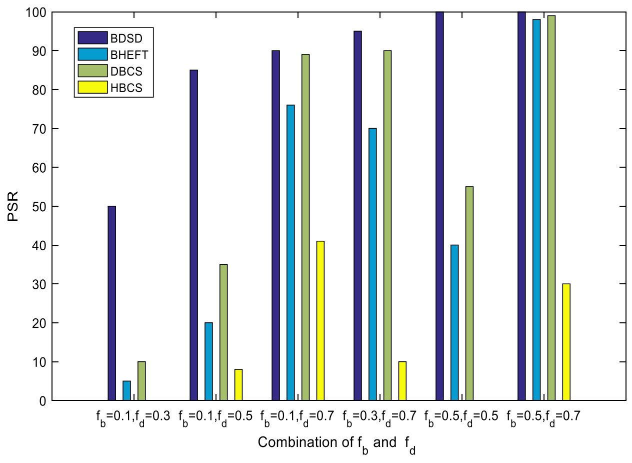

Fig. 10 Randomly generated DAG with 50 nodes, PSR with different constraints

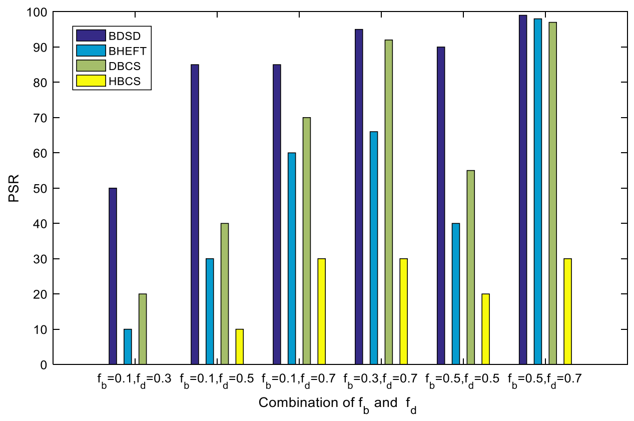

Fig. 11 Randomly generated DAG with 100 nodes, PSR with different constraints

5.3 Time complexity analysis

We assume that a DAG has n tasks, the number of processors is p, and the time complexity of the HEFT algorithm is $\mathrm{O}\left(n^2 \cdot \mathrm{p}\right)$ [14]. The BDSD algorithm includes two stages: task ordering and processor selection. In consideration of time complexity, this algorithm selects either the earliest finish time priority strategy or the cheapest processor. If the insertion policy is selected to calculate the earliest finish time amongst the satisfactory processors, then the complexity is $\mathrm{O}\left(n^2 \cdot p^*\right)$, where p^* refers to the processors that meet the requirements. If the cheapest processor is selected, then the complexity is $\mathrm{O}\left(n^2 \cdot \mathrm{p}\right)$. Each calculation can select only one strategy, so the complexity of BDSD algorithm is $max\left\{\mathrm{O}\left(n^2 \cdot p^*\right), \mathrm{O}\left(n^2 \cdot \mathrm{p}\right)\right\}$. As p^* is smaller than p, therefore the time complexity of the BDSD algorithm is $\mathrm{O}\left(n^2 \cdot \mathrm{p}\right)$.

6 Conclusions

In this paper, we improve a BDSD scheduling algorithm using sub-deadline for workflow under budget and deadline constrained in heterogeneous systems. The scheduling focuses on satisfying cost and time constraints. The proposed algorithm has a low-time complexity. We compare the algorithm to other three algorithms, namely, DBCS, BHEFT and HBCS. The experiments show that BDSD has a high success rate when the time and cost constraints are tight.

In the future, we intend to solve linear programming and obtain a feasible solution for the problem. We also plan to determine a scheduling that minimises makespan under the condition that the total cost of tasks below C.

Acknowledgements This work was supported by Beijing Natural Science Foundation (4162007) and Natural Science Foundation of China (61501008).

References

1. Yu, J., Buyya, R., Ramamohanarao, K.: Workflow scheduling algorithms for grid computing. Metaheuristics Sched. Distrib. Comput. Environ. **146**, 173–214 (2008)
2. Prodan, R., Wieczorek, M.: Bi-criteria scheduling of scientific grid workflows. IEEE Trans. Autom. Sci. Eng. **7**(2), 364–376 (2010)
3. Coffman, E.G., Bruno, J.L.: Computer and Job-shop Scheduling Theory. Wiley, New York (1976)
4. Ullman, J.D.: Np-complete scheduling problems. J. Comput. Syst. Sci. **10**(3), 384–393 (1975)
5. Wu, F., Wu, Q., Tan, Y.: Workflow scheduling in cloud: a survey. J. Supercomput. **71**(9), 3373–3418 (2015)
6. Germain-Renaud, C., Rana, O.: The convergence of clouds, grids, and autonomics. IEEE Internet Comput. **13**(6), 9–9 (2009)
7. Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z.: Online optimization for scheduling preemptable tasks on IaaS cloud systems. J. Parallel Distrib. Comput. **72**(5), 666–677 (2012)
8. Qiu, M., Ming, Z., Li, J., Gai, K., Zong, Z.: Phase-change memory optimization for green cloud with genetic algorithm. IEEE Trans. Comput. **64**(12), 3528–3540 (2015)
9. Gai, K., Qiu, M.K., Zhao, H.: Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing. IEEE Trans. Cloud Comput. **99**, 1–1 (2016)
10. Xu, X.J., Xiao, C.B., Tian, G.Z., Sun, T.: Expansion slot backfill scheduling for concurrent workflows with deadline on heterogeneous resources. Clust. Comput. **20**(1), 471–483 (2017)
11. Arabnejad, H., Barbosa, J., Prodan, R.: Low-time complexity budget-deadline constrained workflow scheduling on heterogeneous resources. Future Gener. Comput. Syst. **55**, 29–40 (2016)
12. Zhang, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control. J. Grid Comput. **11**(4), 633–651 (2013)
13. Arabnejad, H., Barbosa, J.G.: A budget constrained scheduling algorithm for workflow applications. J. Grid Comput. **12**(4), 665–679 (2014)
14. 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)
15. Yu, J., Ramamohanarao, K., Buyya, R.: Deadline/budget-based scheduling of workflows on utility grids. Market-Oriented Grid Util. Comput. **200**(9), 427–450 (2009)
16. Yu, J., Buyya, R., Tham, C.K.: Cost-based scheduling of scientific workflow applications on utility grids. In: First International Conference on e-Science and Grid Computing, IEEE, pp. 1–8 (2005)
17. Yu, J., Buyya, R., Tham, C.K.: QoS-based scheduling of workflow applications on service grids. In: Proceedings of 1st IEEE International Conference-Science and Grid Computing, pp. 5–8 (2005)
18. Casanova, H., Legrand A., Quinson, M.: Simgrid: A Generic Framework for Large-scale Distributed Experiments. In Proceedings of the 10th International Conference on Computer Modeling and Simulation, UKSIM 2008, IEEE, pp. 126–131 (2008)
19. Sakellariou, R., Zhao, H.: A hybrid heuristic for DAG scheduling on heterogeneous systems. Parallel and Distributed Processing Symposium, Proceedings. 18th International. IEEE, vol. 111 (2004)
20. Zheng, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control. J. Grid Comput. **11**(4), 633–651 (2013)
21. Yu, J., Buyya, R.: Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci. Program **14**(3), 217–230 (2006)
22. Wu, Z., Ni, Z., Gu, L., Liu, X.: A revised discrete particle swarm optimization for cloud workflow scheduling. In: Proceedings of 2010 international conference on computational intelligence and security (CIS), IEEE, pp. 184–188 (2010)
23. Rodriguez, M., Buyya, R.: Deadline based resource provisioning and scheduling algorithmfor scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
24. Yuan, Y., Li, X., Wang, Q., Zhang, Y.: Bottom level based heuristic for workflow scheduling in grids. Chin. J. Comput. Chin. **31**(2), 282 (2008)
25. Yuan, Y., Li, X., Wang, Q., Zhu, X.: Deadline division-based heuristic for cost optimization inworkflow scheduling. Inf. Sci. **179**(15), 2562–2575 (2009)

Ting Sun born in 1990, Ph.D. candidate. Her main research interests include cloud computing and combination optimization.

Chuangbai Xiao born in 1962, Ph.D. supervisor and professor. His main research interests include pattern recognition and computer networks.

Xiujie Xu born in 1976, Ph.D. Her main research interests include cloud computing and parallel computing.