IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 8, 2020, accepted April 20, 2020, date of publication May 7, 2020, date of current version May 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2993069

Aggregation Measure Factor-Based Workflow Application Scheduling in Heterogeneous Environments

TING SUN[1], YAQIN ZHANG[1], KAIQI XIONG[2,3], AND CHUANGBAI XIAO[1]
[1]Faculty of Information Technology, Beijing University of Technology, Beijing 100124, China
[2]Department of Mathematics and Statistics, University of South Florida, Tampa, FL 33620, USA
[3]Department of Electrical Engineering, University of South Florida, Tampa, FL 33620, USA

Corresponding author: Chuangbai Xiao (cbxiao@ bjut.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102302, and in part by the National Natural Science Foundation of China under Grant 61971014 and Grant 11675199.

ABSTRACT With the development of heterogeneous distributed computing environment, workflow application scheduling has become an important and challenging problem while Quality of Service (QoS) guarantees are ensured for science workflows. In this paper, we first introduce an aggregation measure factor to balance the execution time and cost of workflow applications. Then, we propose an aggregation measure factor-based scheduling algorithm (AFSA) for workflow applications in a heterogeneous distributed environment. The proposed algorithm through allocating the sub-budget and sub-deadline for each task to choose available processors takes into account of the budget and deadline aggregation to select the processor for the science workflows. Furthermore, we introduce both a planning success rate and normalized deadline (ND) as performance metrics to evaluate workflow application scheduling algorithms. Furthermore, we use both a randomly generated data set and a real-world workflow data set in our experiments for the performance evaluation. Moreover, our experimental results demonstrate that the proposed AFSA has a higher balance factor and an almost equal or higher planning success rate under different workflow application structures compared to the existing algorithms, BHEFT, HBCS, WMFCO, and DBCS.

INDEX TERMS Workflow scheduling, deadline, budget, aggregation measure factor, balance factor, planning success rate.

I. INTRODUCTION

With a large number of diversified resource sets interconnected by high-speed networks in recent years, high-performance heterogeneous distributed computing environments have been rapidly emerged and developed. Computational grids have been used by researchers from different scientific fields to perform complex scientific applications [1]. Those complex scientific applications can be modeled as a workflow that is defined by a directed acyclic graph (DAG) whose nodes represent the tasks of the applications and edges give the executed direction or order of two tasks. Many users desire to employ computing resources or processors to execute their workflows within the requirements of Quality of Service (QoS). Scheduling the tasks of the workflow applications in computing resources is a complex problem because it considers not only the resources' capacity and the tasks' priority but also a user's QoS requirements or parameters. The workflow scheduling problem is an important research hotpot in distributed computing environments [2]. In general, the scheduling problem with QoS parameters is NP-complete [3]. Our challenge is to propose an efficient approach that satisfies a user's predefined QoS parameters with a low time complexity for a schedule of workflow applications compared to the state of the art.

Many workflow scheduling algorithms have been used to execute workflow applications [4]–[6]. Many studies have considered scheduling workflow applications with the QoS requirements, such as time, cost, and reliability [7]–[9]. Some algorithms are used for single-objective workflow scheduling problems [10]–[13], while the others are concerned with multi-objective workflow scheduling problems [14]–[16].

In this paper, we propose an aggregation measure factor-based scheduling algorithm (AFSA) for workflow applications, where time and cost parameters are simultaneously considered in a computational heterogeneous

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

distributed environment. The heterogeneous computing resources (e.g., heterogeneous processors) are distributed in the different locations of a cloud or multiple clouds. The objective of the proposed algorithm is to find the best scheduling of workflow tasks under task deadline and budget constraints that are predefined by a user or time and cost constraints that are predefined by a resource provider. (As noted, budget and deadline are predefined by users, and cost and time are given by a resource provider based on the users' requirements. They need to be consistent and agreed by the user and the resource provider through negotiation. Thus, for simplicity, when there is no confusion, time and deadline may be interchangeably used, so may cost and budget be, in this paper.) The main contributions of this paper are summarized as follows.

1) We propose a new heuristic algorithm for workflow application scheduling subject to the constraints of the time and cost of science workflows.
2) To consider the balance of time and cost parameters, we use normalized deadline to balance the time and cost of science workflows and use it to select the processors for executing the tasks of science workflows.
3) To understand the performance of our algorithm, we employ both randomly generated graphs and real-world applications and experimentally demonstrate that our proposed algorithm performs well in the both datasets. That is, our experiment results show not only the efficiency and effectiveness of the AFSA algorithm but also its performance better than BHEFT [17], HBCS [18], WMFCO [19], and DBCS [1], where the AFSA algorithm has a higher normalized deadline and an almost equal or higher planning success rate under a variety of workflow applications.

The rest of this paper is organized as follows. We give a brief review of related work in section II. In section III, we formulate the workflow schedule problem. In section IV, we present the proposed algorithm, AFSA. In section V, we give the implementation of AFSA and other existing algorithms with illustrative examples and discussions. Finally, we conclude our discussions and present future work in section VI.

II. RELATED WORK

Recently, a number of researchers considered workflow scheduling [1], [17], [18], [20]. Their scheduling approaches can be divided into two major categories: single objective scheduling and multi-objective scheduling [1]. For the single objective scheduling, most researchers considered the execution time of a workflow as a Quality of Service (QoS) parameter. Topcuoglu *et al.* [10] presented the well-known Heterogeneous Earliest-Finish-Time (HEFT) algorithm that is a list-scheduling heuristics one for the workflow scheduling. Bittencourt *et al.* [11] gave a look-ahead variation of the HEFT algorithm, where they considered the impact of execution time of successor of the tasks on current scheduling decisions. However, the algorithm given in Bittencourt *et al.* [11] has a higher time complexity than HEFT. Arabnejad and Barbosa [12] proposed the Predict Earliest Finish Time (PEFT) algorithm, where they considered the impact of execution time of the current task on the next scheduling decision. At the same time, Arabnejad and Barbosa [12] also considered the impact of the execution time of predeceasing of the task on the current scheduling decision. Maurya and Tripathi [13] evaluated and compared the performance of list task scheduling algorithms for heterogeneous computing systems. All of these studies are based on computing resources in grid or cluster environments. Conversely, other studies have considered the workflow scheduling problem in a cloud environment, which is different from a grid or cluster environment [21]. In a cloud environment, a "pay-as-you-go" model is used, computing resources are in different locations, and user tasks may be distributively processed in different computing resources on different sites. Those factors make the scheduling problem very challenging.

For the multi-objective scheduling problem, many researchers consider two or more QoS parameters, such as time, cost, and reliability. Therefore, the scheduling problem has multiple different objectives, such as minimizing total time, minimizing total cost, and obtaining stable performance. Wu *et al.* [15] gave a classification and comparison of these scheduling algorithms. Minimizing the cost under a deadline constraint and minimizing the makespan under a budget constraint are two widely studied categories in the literature [1], [16]–[18], [22]–[24]. (The definition of makespan is given in Section III-B.) For the workflow scheduling problem which minimizes the total cost under a deadline constraint, Yu and Buyya [16] proposed a genetic algorithm to optimize the cost of task processing with a deadline constraint. They proposed a fitness function that combines the cost with the time to measure the quality of the tasks in a DAG according to the given optimization objective and developed two genetic operators, crossover and mutation, for the scheduling problem. Wu *et al.* [24] presented a Revised Discrete Particle Swarm Optimization (RDPSO) algorithm for workflow scheduling. Their optimization objective is to minimize the cost under deadline constraints. For the problem of scheduling workflows subject to the constraint of a budget, Sakellariou *et al.* [22] presented the LOSS and GAIN approaches. For the LOSS approach, tasks are first assigned according to the HEFT [10] or HBMCT [25] algorithms and then according to the available budget. For the GAIN approach, each task is assigned to the processor with the smallest cost. Zeng *et al.* [23] gave a backtracking algorithm named ScaleStar that selects the higher comparative advantage processor to balance time and cost parameters. Zheng and Sakellariou [17] presented the Budget-constrained Heterogeneous Earliest Finish Time (BHEFT) algorithm, which optimizes the total execution time of a workflow and makes the budget as a constraint. Arabnejad and Barbosa [18] proposed a Heterogeneous Budget Constrained Scheduling (HBCS) algorithm that minimizes the deadline of a workflow and satisfies a user's cost budget. They provided an aggregation weight of worthiness

to choose a processor and also considered the influence of the cost factor during the scheduling. Arabnejad *et al.* [1] extended the HBCS algorithm to the Deadline-Budget Constrained Scheduling (DBCS) algorithm which considered time and cost constraints for QoS-based workflow scheduling. DBCS proposed a quality measure to combine time and cost constraints for each processor.

There are also studies concentrated on optimizing several conflict objectives simultaneously [8], [19], [26]–[32]. Garg *et al.* [26] proposed three meta-scheduling online heuristics strategies, including Min-Min Cost Time Trade-off (MinCTT), Sufferage Cost Time Trade-off (SuffCTT), and Max-Min Cost Time Trade-off (Max-CTT), to minimize overall execution time and cost simultaneously on the basis of a trade-off factor. Lee *et al.* [27] presented an Adaptive Dual-Objective Scheduling (ADOS) algorithm, where they minimized makespan and increased resource utilization simultaneously. Bessai *et al.* [28] gave three pareto algorithms with selection policies: cost-based, time-based, and cost-time-based. Talukder *et al.* [29] presented a strategy using Multi-Objective Differential Evolution (MODE) to satisfy time and cost constraint parameters. Yuan *et al.* [30] presented a heuristic scheduling algorithm to minimize the cost of workflow application subject to the user-defined deadline constraint, where they considered both the task priority phase and the resources select phase. Gao *et al.* [19] proposed an algorithm which is called WMFCO for workflow mapping under deadline constraints to minimize the cost in multi-clouds. Chen and Zhang [31] studied an Ant Colony Optimization (ACO) to schedule workflows with multiple QoS parameters such as reliability, time, and cost in computational grids. Zhou *et al.* [32] presented a novel workflow scheduling algorithm that considers the optimization of cost and makespan of scheduling workflows in IaaS clouds. They designed a fuzzy dominance sort based heterogeneous earliest-finish-time algorithm to find and select the best K solutions in each round of solution generation. Prodan and Wieczorek [8] presented the Dynamic Constraint Algorithm (DCA) based on dynamic programming to address the optimization problem subject to the constraints of execution time and cost parameters.

Proper scheduling can decrease data center energy consumption, service-level agreement (SLA) violations, and increase resource utilization [33]. For data center energy reduction, one of the most efficient methods is dynamic voltage and frequency scaling (DVFS) which changes component voltage and frequency to decrease energy consumption. Many studies focus on scheduling and DVFS. Wu *et al.* [34] considered soft error rates during workflow execution due to increasing chip density with DVFS. They proposed a soft error-aware energy-efficient task scheduling approach for workflow applications. Safari and Khorsand [33] presented a new energy-aware scheduling algorithm that arranges the workflow tasks based on their deadlines, and the execution time of the tasks is extended by the use of DVFS. However, further research is required to take account of deadlines, costs, or other SLA parameters.

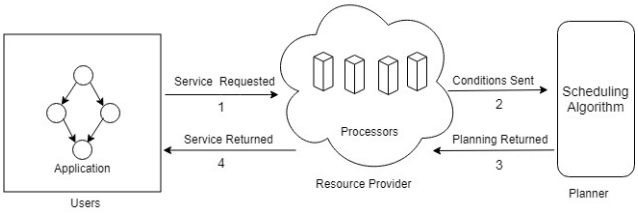

FIGURE 1. Execution of workflow requests.

Faragardi *et al.* [35] 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. Considering the dynamic nature of the workflow changes the budget and the workloads of workflow, Ilyushkin *et al.* [36] proposed a Performance-Feedback Autoscaler (PFA) that is budget-aware and does not consider task execute time estimates for its operation.

In our paper, we present a new scheduling algorithm that considers the time and cost constraints for the scheduling of all tasks. To better understand the performance of our scheduling approach, we compare our algorithm with three well-known algorithms, namely, BHEFT [17], HBCS [18], WMFCO [19], and DBCS [1]. The BHEFT algorithm that is based on the Heterogeneous Earliest Finish Time (HEFT) algorithm [10] and its objective is to minimize the time under a cost constraint. The HBCS algorithm selected a processor with *worthiness* that guaranteed the earliest finish time. The WMFCO algorithm selected the resource under deadline constraint to minimize the cost. The DBCS algorithm used the QoS measure to select the processor that addresses the budget and deadline constraints.

III. PROBLEM DESCRIPTION

In this section, we are going to present the background of this research and give the details of the workflow scheduling problem.

A. BACKGROUND

Figure 1 shows a framework of execution of workflow requests (simply called the workflow scheduling framework), where the workflow scheduling problem needs to be solved. As shown in Figure 1, the workflow scheduling framework consists of the following three parts: Users, Resource Provider, and Planner. When users need workload applications to be done subject to deadline and budget constraints, they transmit a service request to the Resource Provider labeled in 1 given in this figure. The Resource Provider then sends the Users' request with available computing resource and price information, referred as to conditions, to the Planner labeled in 2. After receiving the condition information, the Planner runs the Scheduling Algorithm to find resource processors to match the Users' service requirements and sent

the algorithm output back to the Resource Provider labeled in 3. Furthermore, the Resource Provider informs the Users about the available processors they can use, as labeled in 4. Besides the execution of the Scheduling Algorithm, the Planner can also act as the third party who makes sure a fair deal between the Users and the Resource Provider. The discussion of the fair deal is beyond the scope of this paper.

In this paper, we consider that a resource provider owns computing resources such as processors, each with a price per time unit that is similar to the one in [1]. This research deals with a heterogeneous distributed environment on the cloud. That is, computer processors are not homogeneous; each processor may have a different computational power with a different price. For presentation purposes, we assume that a processor with the high performance (long processing time) requests a higher price, and the processor with a low performance (short processing time) requests a lower price. In order to make the comparison of the results obtained by the scheduling algorithms as in [1], [17], [18], we consider a second as a time unit. Without loss of generality, we assume that each provider has a sufficiently large number of processors as we consider this research in a cloud environment. The goal of a provider is to maximize its profit by executing as many tasks as possible at a time. Therefore, we use the terms, a user, a provider, a processor, and a task, in this paper.

B. WORKFLOW SCHEDULING PROBLEM

A workflow application can be represented as a Directed Acyclic Graph (DAG). A DAG can be modelled by a $G = \{V, E\}$, where a set of nodes, $V = \{v_1, v_2, \ldots, v_n\}$, represents n tasks and a set of directed edges, $E = \{(e_{ij})\}$, stands for data dependencies of those tasks. e_{ij} represents the executed direction from task v_i to task v_j. In other words, the child task can not be executed until all of its parent tasks have been executed and its data has been transferred to the child task.

For each task v_i, let $succ(v_i)$ be a set of all direct successor tasks of v_i and $pred(v_i)$ be a set of all direct predecessor tasks of v_i. In a given DAG, a task with no predecessors is called an entry task and a task with no successors is called an exit task. If there are multiple entry tasks or exit tasks in a DAG, we can add a dummy entry or exit task with zero weight and zero communication edges. Therefore, we will consider the DAG with one entry and one exit task.

Let D is an $n \times n$ matrix of communication data, where D_{ij} be the the size of transmitted data from task v_i to task v_j. The average communication time from task v_i to task v_j is defined as:

$$c_{ij} = \bar{L} + \frac{D_{ij}}{\bar{B}}, \tag{1}$$

where $\bar{L}$ is the average start time of all the processors, and $\bar{B}$ is the average bandwidth among all processor pairs. Let $P = \{p_1, p_2, \ldots p_m\}$ be a set of processors. W is an $n \times m$ computation cost matrix, and $w(v_i, p_j)$ is the execution time of task v_i on processor p_j. Each processor has its own price under unit execution time $R = \{r_1, r_2, \ldots, r_m\}$, and the cost of task v_i on processor p_j is $c(v_i, p_j) = w(v_i, p_j) \times r_j$. The average cost of task v_i ($\bar{C}_i$) is defined as:

$$\bar{C}_i = \frac{\sum_{j=1}^{m} c(v_i, p_j)}{m}. \tag{2}$$

The overall cost for executing an application is defined as:

$$T_c = \sum_{v_i \in V} c(v_i, p_j). \tag{3}$$

The schedule length of a DAG is denoted as a makespan, which can be represented as the finish time of the last task in the DAG defined by:

$$Makespan = AFT(v_{exit}), \tag{4}$$

where $AFT(v_{exit})$ is the actual finished time of the exit task of the DAG.

$EST(v_i, p_j)$ denotes the Earliest Start Time (EST) of a task v_i on a processor p_j, and it is defined as:

$$EST(v_i, p_j) = \max\{T_{avail}(p_j), \max_{v_m \in succ(v_i)} \{AFT(v_m) + c_{mi}\}\}, \tag{5}$$

where $T_{avail}(p_j)$ is the ready time of processor p_j, and c_{mi} is zero if task v_m is assigned to processor p_j. For the entry task v_{entry}, $EST(v_{entry}, p_j) = 0$.

$EFT(v_i, p_j)$ denotes the Earliest Finish Time (EFT) of task v_i on a processor p_j and is defined as:

$$EFT(v_i, p_j) = EST(v_i, p_j) + w(v_i, p_j). \tag{6}$$

We can know that the $EFT(v_i, p_j)$ depends on the earliest start time of task v_i on processor p_j and the execution time of task v_i on processor p_j.

The workflow scheduling problem is to find a scheduling order for the tasks and their corresponding processors that can meet the QoS requirements predefined by users. Specifically, in this paper, we consider the scheduling problem whose objective is to minimize the total of execution time $T_{execution}$ among all given scheduled task orders subject to the total budget to execute the tasks, T_c, less than a predefined budget, B_{pre}, and the makespan less than a predefined deadline, D_{pre}, negotiated by the user and the provider. That is, the total execution time of the workflow should be more than the deadline and the total cost of the workflow executed on processors should not be larger than the budget. The provider hopes to utilize necessary processors to execute the tasks so that the provider can earn as much as possible. We assume that the processor can only execute one task at a time. Mathematically, the scheduling problem in our paper can be formulated as follows.

Find a function $f: V \rightarrow P$ which assigns each task $v_i \in V$ to a processor $p_j \in P$, such that f minimizes $T_{execution}$. That is,

$$argmin_{f \in F} \quad T_{execution}$$

subject to the following two constraints:

$$T_c \leq B_{pre},$$

and

$$Makespan \leq D_{pre},$$

where F is a set of all the task scheduling that assigns tasks to their corresponding processors. For simplicity, B_{pre} and D_{pre} are still called 'budget' and 'deadline' in the following sections.

IV. THE AGGREGATION MEASURE FACTOR-BASED SCHEDULING ALGORITHM

In this section, we present the Aggregation Measure Factor-based Scheduling Algorithm (AFSA) that outputs a schedule, where the algorithm balances both deadline and budget simultaneously. Before describing the algorithm, we will give some definitions used in the algorithm. Processor Utilization Time Rate (UTR) for task v_i on processor p_j is defined as ratio of the actual total execution time of task v_i with the total processor time T_{total}, as shown in equation (7):

$$UTR(v_i, p_j) = \frac{1}{T_{total}} \sum_{i=1}^{k-1} T_{execution}(v_i), \tag{7}$$

where $T_{execution}(v_i)$ is the execution time of task v_i, $T_{total} = deadline \times m$, and m is the number of processors.

The percentage of the average cost (PAC) of task v_i is defined as the ratio between average cost of the current task v_i to the sum of the average costs of the remaining tasks:

$$PAC(v_i) = \frac{\bar{C}_i}{\sum\limits_{v_j \in N} \bar{C}_j}, \tag{8}$$

where $\bar{C}_i$ is the average cost of the task v_i and N is the set of unscheduled tasks.

Our proposed algorithm, AFSA, includes two phrases: task selection and processor selection.

A. THE TASK SELECTION

In this section, we will first present the task selection and then give the processor selection. We use the upward rank ($rank_u$) [11] to prioritize tasks. The $rank_u$ is defined as follows.

$$rank_u(v_i) = \bar{w}(v_i) + \max_{v_m \in succ(v_i)} (c_{im} + rank_u(v_m)), \tag{9}$$

where $\bar{w}(v_i)$ is the average execution time of task v_i over all processors, c_{im} is the average communication time from task v_i to task v_m, $succ(v_i)$ is the set of all direct successors of v_i, and $rank_u$ is the longest execution and communication time from task v_i to the exit task v_{exit}.

B. THE PROCESSOR SELECTION

For the processor selection, we present a new strategy to choose processors. The processors must execute the current task within the budget/cost and deadline/time constraints, where the budget and the deadline are predefined by the Users, and the cost and the time are given by the Resource Provider. In the following paragraphs, we give detailed discussions. According to [20], remaining budget (RB) can be calculated by

$$RB = B - \sum_{i=1}^{k-1} c_i, \tag{10}$$

where B is the given budget and c_i is the reservation cost of the allocated task v_i. The expected remaining budget (ERB) for task v_i can be calculated by

$$ERB(v_i) = \frac{RB}{l}, \tag{11}$$

where l is the number of unscheduled tasks.

The sub-budget of the current task v_k can be calculated by:

$$SBC(v_k) = \bar{C}_k + ERB(v_k) \times PAC(v_k) \tag{12}$$

By calculating the sub-budget for each task, we can find processors which are satisfied

$$C_{k,m} \leq SBC(v_k). \tag{13}$$

Using equation (13), we can find the processor set P' which satisfy the budget constraint. After obtaining the set of processors which satisfy the budget constraint, we will continue to consider the deadline constraint. We will present the definition of the sub-deadline of a task. According to [1], the sub-deadline of the current task, as shown by equation:

$$SD(v_i) = \min_{v_j \in succ(v_i)} \{SD(v_j) - c_{ij} - \min_{p_m \in Q^*} w(v_i, p_m)\}, \tag{14}$$

where $succ(v_i)$ is a set of all direct successor tasks of v_i, c_{ij} is the average communication time from task v_i to task v_j, and $w(v_i, p_m)$ is the execution time of task v_i on processor p_m. For the exit task v_{exit}, the sub-deadline is equal to the user's deadline, $SD(v_{exit}) = D$.

For processor selection, we consider the two QoS parameters, UTR and cost, to obtain the available processors, where budget and deadline constraints are balanced. To balance the two constraints, for each task v_i, we define *the aggregation measure factor* (AGG) for processor $p_m \in Q^*$ as follows:

$$AGG(v_i, p_m) = UTR(v_i, p_m) \times c(v_i, p_m). \tag{15}$$

In other words, $AGG(v_i, p_m)$ represents the degree of balance for the processor utilization time rate and cost parameters of task v_i executed on processor p_m.

We also consider the sub-deadline of each task affect the selection. Let us define p^* by:

$$p^* = \operatorname*{argmin}_{p_m \in Q^*} AGG(v_i, p_m).$$

Thus, we consider the most suitable processor for task v_i as follows.

$$p_{sel} = \begin{cases} \underset{p_m \in Q^*}{\operatorname{argmin}} AFT(v_i, p_m), & AFT(v_i, p^*) > SD(v_i) \\ p^*, & AFT(v_i, p^*) \leq SD(v_i). \end{cases} \quad (16)$$

If task v_i has a higher finish time on processor p^* than its sub-deadline, we will choose the minimum finish time processor as the most suitable processor for task v_i. Otherwise, we will choose p^* as the most suitable processor for task v_i.

Algorithm 1 presents the Aggregation Measure Factor-based Scheduling Algorithm (AFSA). First, we sort the tasks of the DAG in lines 1 and 2. After obtaining the highest priority task, we calculate the sub-budget and sub-deadline in lines 4 to 7. Then, in lines 8 to 10, we select processors that satisfy the budget constraint. Furthermore, we find the processor with the *AGG* given in (15) in lines 12 to 14. Finally, we select the processor for task v_i using equation (16) in line 15 and update the remaining budget in line 16.

1) THE TIME COMPLEXITY

Now we consider the time complexity of AFSA. Line 1 of AFSA calculates the $rank_u$ value for each task. Suppose the number of tasks in the workflow is n and the number of processors is p. Then, the time complexity of line 1 is $O(n \times p)$. Form lines 5 to 7, AFSA calculates the remaining budget, sub-budget, and sub-deadline for each task. Its time complexity is $O(np)$. Lines 8 to 11 of the AFSA algorithm finds the suitable processors that satisfy the sub-budget, and their time complexity is $O(p)$. Lines 12 to 14 of the AFSA algorithm to calculate the *AGG* measure with $O(p^*)$, where $|p^*| < |p|$. Line 15 selects the processor with Equation (16) and the time complexity is $O(1)$. Line 16 calculates the remaining budget ERB and its time complexity is $O(1)$. Thus, the time complexity from lines 3 to 17 is $O(n(np + p + p^* + 1))$, which is equivalent to $O(n^2 p)$. Therefore, the total time complexity of the AFSA is $O(n^2 p + np)$, which is equivalent to $O(n^2 p)$.

2) AN ILLUSTRATIVE EXAMPLE

Let us consider a 10-task workflow application [10] whose DGA is depicted in Figure 2. In this figure, the data on each edge represents the average communication time between two tasks. We make this DAG executed in three processors with different computational abilities. Furthermore, the average execution time $w(v_i, p_j)$ of task v_i on processor p_j and the execution cost $c(v_i, p_j)$ of task v_i on processor p_j are given in Tables 9 and 3, respectively. Assume that the user request the deadline is 100 and the budget is 130 for the DAG. Based on these data, First we use Equation (9) to obtain the non-ascending order of tasks: $v_1, v_4, v_3, v_2, v_5, v_6, v_9, v_7, v_8, v_{10}$. Then, we calculate the expected remaining budget, processor set P', the sub-deadline, the Earliest Start Time, and the Earliest Finish Time by using the AFSA algorithm to find the scheduling for each task as shown in Table 4.

Figure 3 shows the scheduling plan by AFSA using DAG in Figure 2. The makespan of this scheduling is 85 that satisfies the deadline 100. The cost of this scheduling is 55.36 that satisfies budget 130. The makespan of HEFT is 80; however, the cost is 59.81. The makespan of AFSA is bigger 5 units than the makespan of HEFT, the total cost of AFSA has reduced 4.45 cost units.

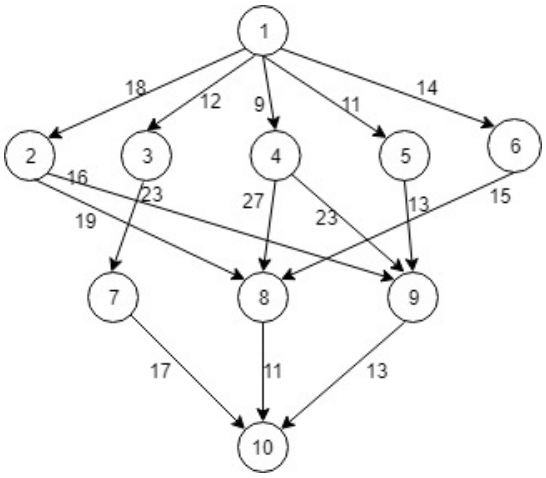

FIGURE 2. An example DAG from [10].

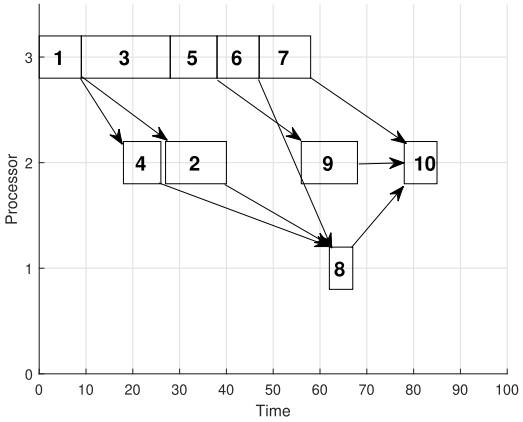

FIGURE 3. The scheduling plan generated by AFSA using DAG in Figure 2.

TABLE 1. The average execution time of tasks on processors (sec).

Task	p_1	p_2	p_3
v_1	14	16	9
v_2	13	19	18
v_3	11	13	19
v_4	13	8	17
v_5	12	13	10
v_6	13	16	9
v_7	7	15	11
v_8	5	11	14
v_9	18	12	20
v_{10}	21	7	16

TABLE 2. The price used to run a task on different processors.

Processor	Price
p_1	0.91
p_2	0.52
p_3	0.43

TABLE 3. The computation cost of tasks on processors per sec.

Task	p_1	p_2	p_3
v_1	12.74	8.32	3.87
v_2	11.83	9.88	7.74
v_3	10.01	6.76	8.17
v_4	11.83	4.16	7.31
v_5	10.92	6.76	4.30
v_6	11.83	8.32	3.87
v_7	6.37	7.8	4.73
v_8	4.55	5.72	6.02
v_9	16.38	6.24	8.60
v_{10}	19.11	3.64	6.88

TABLE 4. AFSA data for the DAG in Figure 2.

Task	$rank_u$	SBC	P'	SD	$p*$	EST	AFT	cost
v_1	108.00	13.24	p_1,p_2,p_3	5.00	p_3	0	9	3.87
v_4	80.01	12.01	p_1,p_2,p_3	32.67	p_2	18	26	4.16
v_3	80.00	12.43	p_1,p_2,p_3	34.33	p_3	9	28	8.17
v_2	77.01	13.69	p_1,p_2,p_3	39.67	p_2	27	40	11.83
v_5	69.01	9.16	p_2, p_3	42.67	p_3	28	38	4.30
v_6	63.34	9.58	p_2,p_3	49.33	p_3	38	47	3.87
v_9	44.34	11.96	p_2, p_3	72.33	p_2	56	68	6.24
v_7	42.67	6.76	p_1, p_3	68.33	p_3	47	58	4.73
v_8	35.67	5.52	p_1	74.33	p_1	62	67	4.55
v_{10}	14.67	9.87	p_2, p_3	100	p_2	78	85	3.64
Total								55.36

V. PERFORMANCE EVALUATION

In this section, we compare the AFSA algorithm with the BHEFT [17], HBCS [18], WMFCO [19] and DBCS [1] algorithms. We consider both randomly generated and real-application workflows to evaluate the algorithms. The simulation experiments for the evaluation were performed on MATLAB using a Windows 10 machine which has the following specifications: quad-core Intel i5-7200U CPU @ 2.70 GHz with 8GB DIMMs.

A. PERFORMANCE MATRIX

In our experiment, we consider two performance metrics to evaluate and compare our algorithm with other algorithms.

We adopt the planning success rate (PSR) to evaluate the algorithm defined by Arabnejad *et al.* [1]. The PSR is defined as follows:

$$PSR = 100 \times \frac{N_{success}}{N_{total}}, \tag{17}$$

where $N_{success}$ is the number of schedules which satisfied the user's budget and deadline constraints, and N_{total} is the total number of the schedules.

To further evaluate the performance of different algorithms, we use the normalized deadline by [37]:

$$ND = \frac{M_{alg}}{M_{min}}, \tag{18}$$

where M_{alg} is the makespan of an algorithm, M_{min} is the minimum makespan of the workflow uesd to execute all tasks on the fastest processors without the consideration of the cost constraint. It is easy to see that the value of ND not less than 1. When ND is close to 1, we can know that the scheduling makespan of the algorithm is close to the minimum makespan of the workflow.

Algorithm 1 Aggregation Measure Factor-Based Scheduling Algorithm (AFSA)

```
Calculate all tasks rank_u using Equation (9)
R ← Set of all tasks with the non-ascending order of rank_u
while R ≠ ∅ do
    v_i = the ready task with the highest rank_u value
    calculate the task remaining budget for task v_i according to Equation (10)
    calculate the sub-budget of task v_i according to Equation (12)
        calculate the sub-deadline of task v_i according to Equation (14)
    for all p_j ∈ P
      if c(v_i, p_j) ≤ SBC(v_i)
        insert p_j into P*
    end for
    for all p_m ∈ P*
      compute AGG(v_i, p_m) using Equation (15)
    end for
select p_sel for task v_i according to Equation (16)
update the remaining budget ERB according to Equation (11)
end while
return Schedule Map
```

B. RESULTS FOR RANDOMLY GENERATED WORKFLOWS

1) WORKFLOW STRUCTURE AND DATASETS

The synthesis task graph generator [38] generates a DAG structure that has five parameters: n, *fat*, *regularity*, *density*, and *jump*, where n is the number of nodes in the DAG, *fat* affects the height and width of the DAG, *regularity* represents the consistency of the number of nodes in each level, *density* is the number of edges between two levels of the DAG, and it indicates the data dependencies between different layers of tasks, and *jump* is the maximum number of layers that can be spanned transfer between different tasks. A *jump* means an edge can go from level l to level $l + jump$. In our experiment, for the random DAG workflow, we consider the DAG with different parameters shown in Table 5. With these parameters, each DAG is generated by choosing one value of each parameter randomly from the parameter data set. The total number of random DAGs generated in our experiment is 3750. For each DAG, the data amount transmitted between tasks is randomly generated according to the communication to computing ratio (ccr). A larger ccr means the DAG has a dense communication. Otherwise, the DAG is computationally intensive. The range of values that we used in our simulation was [0.5, 1, 1.5, 2, 2.5, 3] for ccr.

We consider two sites that include multiple clusters consisting of heterogeneous processors, as discussed in [1].

TABLE 5. Parameters setting in the random DAG.

parameter	value
n	20, 40, 60, 80, 100, 120
fat	0.1, 0.3, 0.5, 0.7, 0.9
$regularity$	0.1, 0.3, 0.5, 0.7, 0.9
$density$	0.1, 0.3, 0.5, 0.7, 0.9
$jump$	1, 2, 3, 4, 5

We use the Sophia and Lille sites, where the Sophia site has a larger number of slow speed processors and the Lille site has a larger number of fast speed processors. Table 6 gives the configurations of clusters at two sites, where the number of processors used, the processor speed in GFlop/s, and processor cost are given in each cluster within its site. As defined in [17], the diverse price of the heterogeneous processor is normalized as follows. Let β_j be the ratio of p_j processing capacity to the fastest processor capacity, and the price of processor p_j be normalized as $price(p_j) = \frac{\beta_j(1+\beta_j)}{2}$, which is belong to (0, 1].

2) DEADLINE AND BUDGET CONSTRAINTS

To have better understanding of the performance of our proposed algorithm and its corresponding algorithms that are used for comparison in the research, we consider the different values of budget and deadline, and the users pre-define those values in their QoS requirement. As mentioned before, B_{pre} and D_{pre} are predefined by the user. In our simulation, in order to consider how the different values of B_{pre} and D_{pre} impact on the performance of the algorithms under study, we select the values of B_{pre} and D_{pre} in the following ways [17]:

$$D_{pre} = M_{min} + \phi_d(M_{max} - M_{min}). \tag{19}$$

where M_{min} is the makespan of the HEFT algorithm and M_{max} is the three times of the makespan of the HEFT algorithm. ϕ_d is a parameter to control the range of the value of D_{pre} so that we can understand how different values of D_{pre} impact on the performance of those algorithms.

Furthermore, for each DAG, we calculated the maximum cost (C_{max}) and minimum cost (C_{min}) of all tasks that were executed on the processors as the highest and lowest values. For the current DAG, the budget constraint is defined by:

$$B_{pre} = C_{min} + \phi_b(C_{max} - C_{min}), \tag{20}$$

Similar to ϕ_d, ϕ_b is a parameter to control the range of the value of B_{pre} so that we can understand how different values of B_{pre} impact on the performance of those algorithms. Budget B_{pre} and deadline D_{pre} are called "tight" if their values are relatively small. Otherwise, they are called "loose." Note that those values are considered relatively small or large by compared to the makespan of the HEFT algorithm, M_{min}, and minimum cost, C_{min}, and they are determined by the user and the provider based on the user's application. In our simulation, we have found when the budget and deadline constraints are loose, the DBCS, HBCS, BHEFT, and AFSA algorithms almost have the same PSR value. To have a sensitive comparison, we have chosen the small ranges of parameters ϕ_b and ϕ_d.

Here, we choose ϕ_d, $\phi_b = 0.1, 0.2, 0.3$, respectively.

TABLE 6. The cluster description in the experiment.

site	cluster	Processor	Power(GFlop/s)	price
Sophia	sol	16	8.938e9	0.19
Sophia	helios	24	7.731e9	0.16
Lille	chinqchint	20	22.270e9	0.64
Lille	chimint	16	23.531e9	0.70

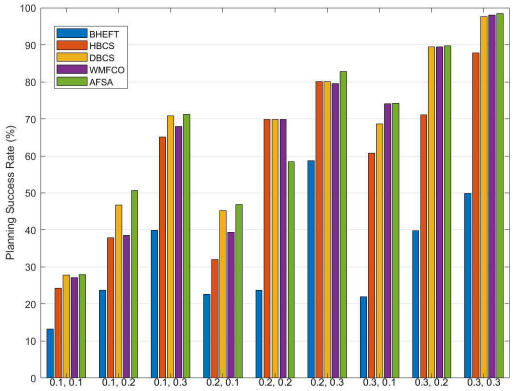

FIGURE 4. PSR versus deadline and budget constraints on the Lille site.

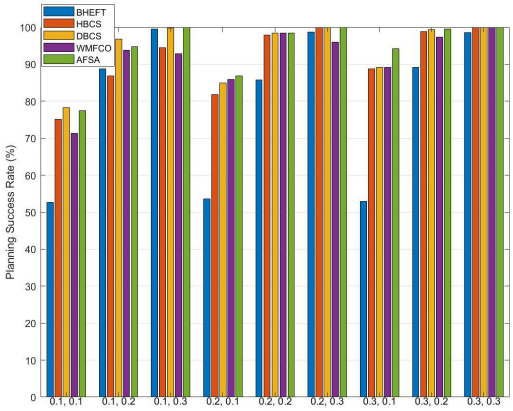

FIGURE 5. PSR versus deadline and budget constraints on the Sophia site.

3) RESULTS AND ANALYSIS

In this subsection, we compare the AFSA algorithm with the HBEFT algorithm [17], the HBCS algorithm [18], the WMFCO algorithm [19], and the DBCS algorithm [1] under the randomly generated DAGs. Figures 4 and 5 show the PSR of the five algorithms with different deadline and budget constraints on the two sites with different price processors. Figure 4 shows the PSR obtained by the five algorithms on the Lille site. From this figure, we can see the WMFCO algorithm obtains PSR values close to the ones obtained by the AFSA algorithm when the constraints are loose. When the constraints are tight, AFSA obtains better results than WMFCO. The AFSA algorithm has a better PSR compared to the HBCS

algorithm, DBCS algorithm, and BHEFT algorithm. The PSR of DBCS is almost close to that of the HBCS algorithm. The HBEFT algorithm has the lowest PSR compared to the other four algorithms.

Figure 5 presents the average PSR of the five algorithms with different deadline and budget constraints in Sophia site. As shown in Figure 5, when the $\phi_d = 0.1$, the PSR of the AFSA algorithm is almost the same for the DBCS algorithm, whereas the BHEFT and HBCS algorithms are less than the AFSA and DBCS algorithms. As the ϕ_b increases, the PSR of the four algorithms increases. When $\phi_d = 0.3$, $\phi_b = 0.1$, the AFSA algorithm has the highest PSR compared to the DBCS, HBCS, and BHEFT algorithms. The AFSA algorithm has a higher PSR value compare to the WMFCO algorithm when the deadline and budget constraints are small. As the deadline and budget increases, the WMFCO algorithm has the same PSR value as the AFSA algorithm. That is, the ASFA algorithm has the best performance under the tight deadline and budget among all the five algorithms. By comparing the two sites of the PSR, we have seen that the PSR values on Lille are lower than the ones on the Sophia site. On the Lille site, BHEFT has a lower PSR compared to DBCS, HBCS, WMFCO, and AFSA. DBCS, HBCS, WMFCO, and AFSA achieve the same PSR values on Sophie site when the constraints are loose.

In this paper, in order to better understand how the workflow application structures make an impact on the algorithm, we analyze how the DAG structure parameters affect PSR when the HBEFT, DBCA, HBCS, WMFCO, and AFSA algorithms are used on the Sophia site. We observed the different values of the average Planning Successful Rate (PSR) via the five algorithms under the different setup of n, *fat*, *regularity*, *density*, *jump*, and *ccr*, where we consider the change of one parameter but the unchanged of the other parameters in our experiments.

Figure 6 shows the PSR values by varying the number of tasks n when the other parameters are fixed, where $fat = 0.7$, $regularity = 0.3$, $density = 0.7$, $jump = 1$, $ccr = 2$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As n increases from 20 to 120, the AFSA algorithm has almost the same PSR as the HBCS, WMFCO, and DBCS algorithms, whereas the BHEFT algorithm has a lower PSR than the other four algorithms.

Figure 7 depicts the PSR by varying the *fat*, but the other parameters are fixed, where $n = 60$, $regularity = 0.3$, $density = 0.7$, $jump = 1$, $ccr = 2$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As shown in Figure 7, when $fat = 0.1$, the PSR is almost zero for all the five algorithms. However, as *fat* increases from 0.1 to 0.5, the PSR of the AFSA algorithm, the HBCS algorithm, the WMFCO algorithm, and the DBCS algorithm increases dramatically, whereas the PSR obtained by the BHEFT algorithm increases relatively slowly. Furthermore, when the *fat* is more than 0.5, the PSR nearly reaches 100% for all the five algorithms: BDBC, HBCS, WMFCO, and DBCS. However, when the *fat* is more than 0.5, the PSR of the BHEFT algorithm is about 80%, which is what we expect. The value of parameter *fat* affects the parallelism of DAG tasks.

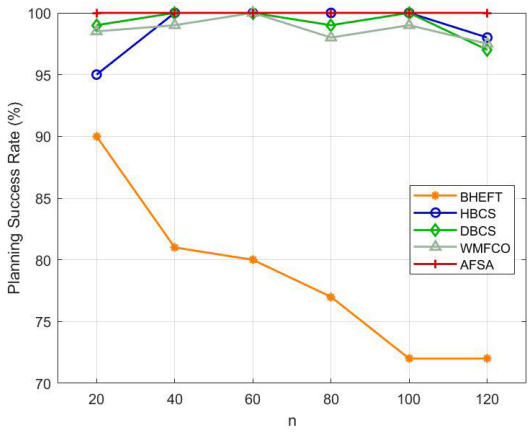

FIGURE 6. PSR versus n.

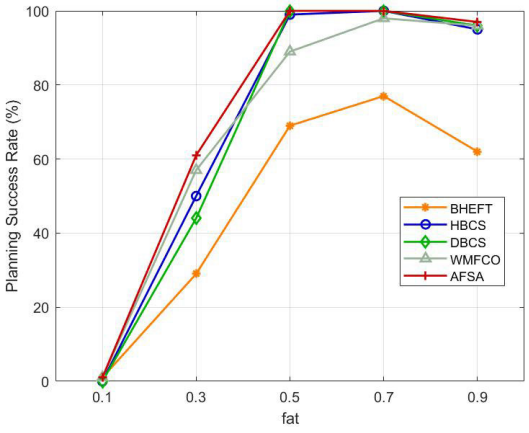

FIGURE 7. PSR versus fat.

The larger the *fat*, the higher the parallelism of tasks. When the parallelism of DAG tasks is higher, the PSR increases more. Hence, we can conclude that the PSR increases when the *fat* increases. For this reason, we can give the users a recommendation: if the user's workflow application has a higher value of *fat*, she/he can loose the restriction of the deadline and give more time to the resource provider.

Figure 8 shows the PSR by varying the *regularity* but the other parameters are fixed, where $n = 60$, $fat = 0.7$, $density = 0.7$, $jump = 1$, $ccr = 2$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As *regularity* increases from 0.1 to 0.9, the PSR of the AFSA algorithm is 100%, the PSR values of the HBCS and DBCS algorithms are between 90% and 100%, the PSR of WMFCO algorithm is close to the one obtained by the DBCS algorithm; it is between 95% and 100%, whereas the PSR of the BHEFT algorithm ranges from 70% to 80%. From this result, we can know that the AFSA algorithm has a higher PSR compared with the other four algorithms.

Figure 9 shows the PSR by varying the *density*, but the other parameters are fixed, where $n = 60$, $fat = 0.7$, $regularity = 0.3$, $jump = 1$, $ccr = 2$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As shown in Figure 9, when $density = 0.1$, the PSR is less than 70% for all the five algorithms. However, as *density* increases from 0.1 to 0.3, the PSR of the AFSA

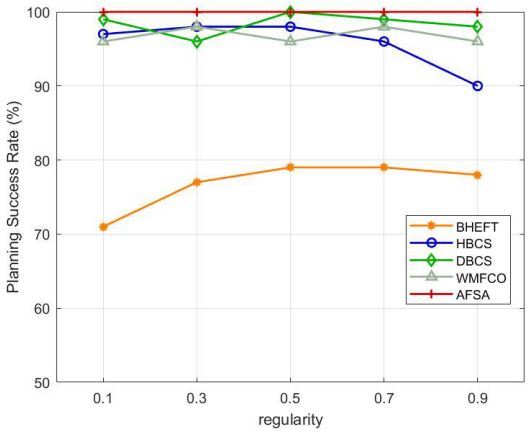

FIGURE 8. PSR versus regularity.

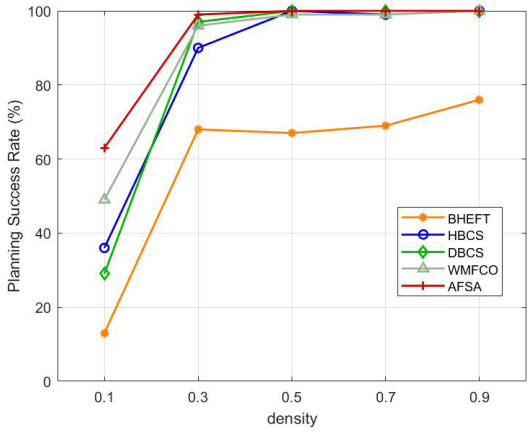

FIGURE 9. PSR versus density.

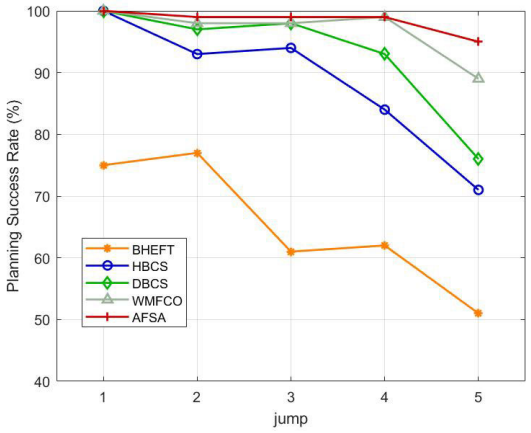

FIGURE 10. PSR versus jump.

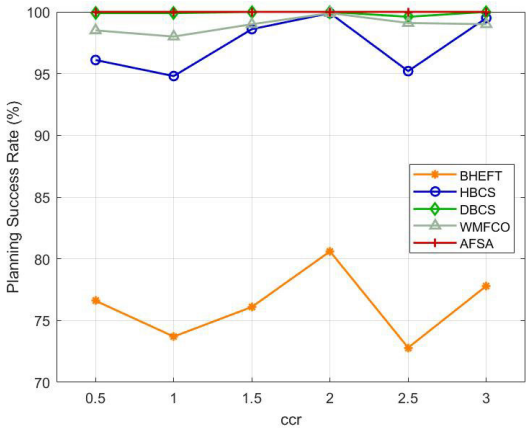

FIGURE 11. PSR versus ccr from 0.5 to 3.

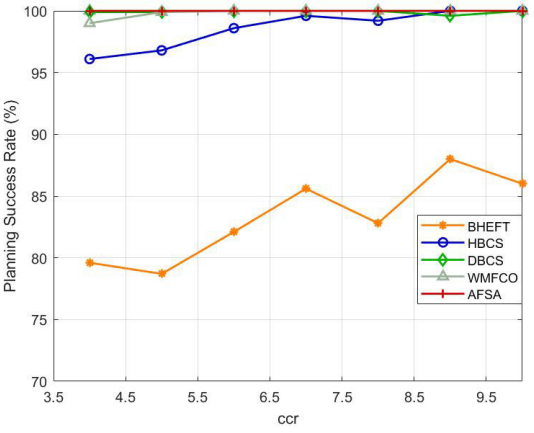

FIGURE 12. PSR versus ccr from 4 to 10.

algorithm, the HBCS algorithm, the WMFCO algorithm, and the DBCS algorithm increase dramatically, whereas the PSR obtained by the BHEFT algorithm increase relatively slowly. When *density* is more than 0.3, the PSR nearly reaches 100% for AFSA, HBCS, WMFCO, and DBCS, the PSR of BHEFT is about 70%. Figure 10 shows the PSR by varying the *jump*, but the other parameters are fixed, where $n = 60$, $fat = 0.7$, $regularity = 0.3$, $density = 0.7$, $ccr = 2$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As *jump* increases from 1 to 5, we can see that the PSR of the five algorithms decreases. The WMFCO algorithm has the same PSR as the AFSA algorithm when $jump = 4$. The AFSA algorithm has a higher PSR compared to the HBCS, DBCS, and BHEFT algorithms.

Figure 11 shows the PSR by varying the *ccr*, but the other parameters are fixed, where $n = 60$, $fat = 0.7$, $regularity = 0.3$, $density = 0.7$, $jump = 1$, $\phi_d = 0.1$, and $\phi_b = 0.1$. As *ccr* increases from 0.5 to 3, the AFSA algorithm has a higher PSR compared to the HBCS, WMFCO, and BHEFT algorithms. AFSA has the same PSR as DBCS. Figure 12 shows the PSR by varying the *ccr* from 4 to 10. From this figure, we can see that the AFSA algorithm has a higher PSR than the HBCS and BHEFT algorithms. AFSA has the same PSR as DBCS and WMFCO. Compared to Figure 11, Figure 12 shows that the PSR of the BHEFT algorithm increases as the ccr increases.

To further evaluate the performance of the algorithms, we compare the ND with different algorithms for random workflow applications. We generated 100 sample workflow application, each contains 100 tasks with $fat = 0.5$, $regularity = 0.5$, $jump = 2$, $density = 0.7$, and $ccr = 2$. The experiment use 16 processors to evaluate. We used these five algorithms for scheduling with different budget parameters and recorded their average normalized deadline.

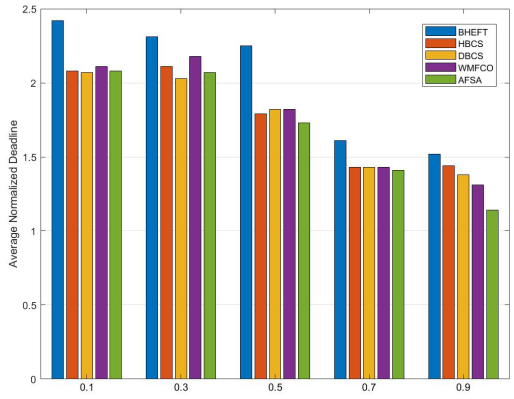

FIGURE 13. Average normalized deadline versus budget constraint on the Lille site.

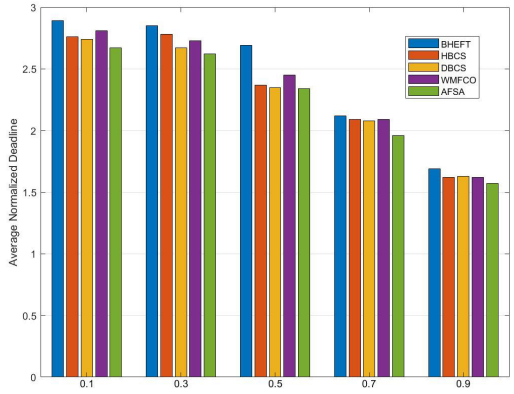

FIGURE 14. Average normalized deadline versus budget constraint on the Sophia site.

Figures 13 and 14 show the average normalized deadlines with different budget parameters on the Lille and Sophia sites, respectively. As shown in Figure 13, AFSA has a lower normalized makespan and BHEFT has the highest normalized deadline compared to the other algorithms when $\phi_b = 0.1$, $\phi_b = 0.3$, and $\phi_b = 0.5$. When ϕ_b is increased to 0.7 and 0.9, the average normalized makespan decreases, where AFSA and DBCS have the same values. As shown in Figure 14, BHEFT and WMFCO have a higher average normalized makespan compared to other algorithms, when $\phi_b = 0.1$. When $\phi_b = 0.9$, we see that BHEFT, HBCS, and DBCS almost have the same average normalized makespan; AFSA has a lower average of ND compared to the other four algorithms. By comparing Figure 13 with Figure 14, we can see that the Sophia site has a higher average normalized makespan than the Lille site.

C. RESULTS FOR REAL WORLD WORKFLOWS

To further evaluate the algorithms in the real-world workflow applications, we choose three well-known application structures [39], namely, Montage, LIGO Inspiral, and Epigenomics. The Montage is an application that can be executed in grid environments and utilizes the image to generate custom mosaics of the sky. In the experiments, we considered the Montage structure with 25, 50, and 98 tasks. The LIGO Inspiral workflow is used to analyze the data from the coalescing of compact binary systems. We considered the LIGO inspiral structure with 30, 50, and 120 tasks. Epigenomics is a highly parallel workflow with multiple tasks that are run on independent chunks of data in parallel. We considered the Epigenomics structure with 24 and 100 tasks. We repeatedly run the simulation 1000 times to obtain the results given in Tables 7, 8, and 10, respectively.

In Table 7, we show the PSR values obtained by varying *ccr* on 16 processors with the Montage applications, where the above five algorithms are used. When $ccr = 0.25$, for 25 tasks of Montage workflow, the AFSA algorithm improves 12.5% of PSR compared with the BHEFT algorithm; conversely, for 50 tasks of Montage workflow, the AFSA algorithm improves 23.9% of PSR compared with the BHEFT algorithm. When $ccr = 2$, for 25 tasks of Montage workflow, the AFSA algorithm improves 9.7% of PSR compared with the DBCS algorithm; For 50 tasks of the Montage workflow, the AFSA algorithm improves 6.2% of the PSR value compared to the BHEFT algorithm. For 98 tasks of the Montage workflow in Table 9, AFSA improves 27.4 of the PSR value compared to BHEFT when $ccr = 0.25$; BHEFT has the lowest PSR compared to the other four algorithms. Table 8 presents the PSR when we vary *ccr* on 16 processors with the LOGO Inspiral applications by using the above five algorithms. As shown in Table 8, the AFSA algorithm is better than the DBCS, HBCS algorithms and significantly better than the BHEFT algorithm for Inspiral applications with 30 and 50 tasks, respectively. Table 9 shows that for Inspiral applications with 120 tasks, AFSA has a higher PSR than BHEFT. AFSA, WMFCO, DBCS, and HBCS almost have the same PSR when $ccr = 0.5$.

Table 10 shows the PSR values when we vary *ccr* on 16 processors with the Epigenomics applications based on the five algorithms. As shown in Table 10, the AFSA algorithm is better than the DBCS and HBCS algorithms; when $ccr = 0.25$, AFSA improves 15% and 19.6% of the PSR value compared to BHEFT for 24 tasks and 100 tasks, respectively. The AFSA algorithm has lower performance when $ccr = 2$. However, the AFSA algorithm has better performances than the BHEFT algorithm. As shown in Table 10, when $ccr = 0.25$ and $ccr = 0.5$, we see that the PSR values of the BHEFT algorithm are 76.3% and 78.9%, whereas the PSR of AFSA are 95.2% and 93.2%.

The above experimental results have shown that the AFSA algorithm achieves the best performance compared to the DBCS, HBCS, WMFCO, and BHEFT algorithms when different *ccr* values are chosen. The results also indicate that when *ccr* increases, the PSR of the five algorithms decreases. This is an expected result as the value of *ccr* is reduced, which implies that we have the more feasible to find a schedule for the same budget and deadline.

To further evaluate the performance of the algorithm, we also compare the PSR of our algorithm with the PSR of the DBCS, HBCS, BHEFT, and WMFCO algorithms. Figures 15-20 show the PSR values of the five algorithms

TABLE 7. The PSR of the five algorithms on Montage with 25 and 50 tasks.

DAG	Montage (25)					Montage (50)				
Algorithm	AFSA	WMFCO	DBCS	HBCS	BHEFT	AFSA	WMFCO	DBCS	HBCS	BHEFT
$ccr = 0.25$	94.6%	91.8%	93.6%	92.0%	82.1%	95.7%	92.1%	93.3%	92.7.2%	61.8%
$ccr = 0.5$	90.5%	90.4%	87.2%	86.8%	78.0%	96.2%	92.3%	93.4%	93.6%	63.9%
$ccr = 1$	83.3%	78.8%	79.6%	78.7%	73.5%	86.6%	83.5%	83.3%	85.6%	74.5%
$ccr = 2$	77.6%	66.8%	65.9%	65.4%	68.9%	73.6%	73.4.5%	72.9%	73.1%	67.4%

TABLE 8. The PSR of the five algorithms on Inspiral with 30 and 50 tasks.

DAG	Inspiral (30)					Inspiral (50)				
Algorithm	AFSA	WMFCO	DBCS	HBCS	BHEFT	AFSA	WMFCO	DBCS	HBCS	BHEFT
$ccr = 0.25$	92.2%	91.6%	92.1%	91.4%	78.9%	94.1%	92.6%	91.7%	91.4%	71.3%
$ccr = 0.5$	91.7%	90.3%	91.1%	91.7%	75.1%	96.3%	95.3%	95.6%	94.7%	70.4%
$ccr = 1$	93.5%	93.1%	91.2%	90.5%	75.1%	96.7%	95.7%	96.1%	94.9%	72.5%
$ccr = 2$	89.5%	88.3%	90.6%	86.3%	68.8%	94.3%	93.8%	94.2%	93.8%	72.4%

TABLE 9. The PSR of the five algorithms on Montage with 98 tasks and Inspiral with 120 tasks.

DAG	Montage (98)					Inspiral (120)				
Algorithm	AFSA	WMFCO	DBCS	HBCS	BHEFT	AFSA	WMFCO	DBCS	HBCS	BHEFT
$ccr = 0.25$	94.5%	93.2%	91.6%	92.1%	67.1%	95.2%	91.2%	90.2%	86.3%	77.3%
$ccr = 0.5$	90.2%	90.2%	88.2%	89.1%	62.3%	93.2%	92.1%	93.2%	91.6%	70.1%
$ccr = 1$	86.2%	84.3%	84.2%	83.6%	57.7%	96.7%	94.9%	93.5%	93.4%	64.7%
$ccr = 2$	76.8%	76.4%	76.2%	77.5%	50.9%	97.0%	95.4%	94.7%	94.2%	63.8%

TABLE 10. The PSR of the five algorithms on Epigenomics with 24 and 100 tasks.

DAG	Epigenomics (24)					Epigenomics (100)				
Algorithm	AFSA	WMFCO	DBCS	HBCS	BHEFT	AFSA	WMFCO	DBCS	HBCS	BHEFT
$ccr = 0.25$	98.2%	97.1%	95.5%	94.9%	83.2%	95.9%	93.9%	93.2%	94.5%	76.3%
$ccr = 0.5$	94.2%	93.2%	92.6%	93.1.7%	84.6%	92.5%	90.1%	90.2%	93.7%	78.9%
$ccr = 1$	85.6%	83.3%	79.5%	80.6%	66.8%	91.3%	90.6%	87.3%	90.5%	76.9%
$ccr = 2$	79.2%	79.2%	66.8%	64.2%	62.1%	78.3%	72.9%	74.1%	73.2%	69.1%

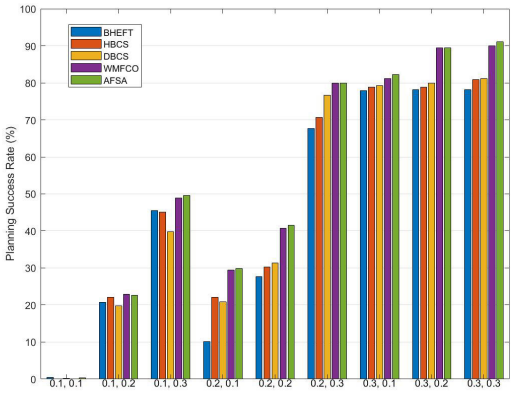

FIGURE 15. The PSR for Montage on the Lille site.

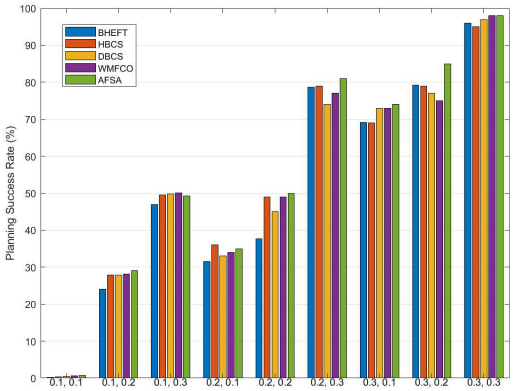

FIGURE 16. The PSR for Montage on the Sophia site.

for the Montage, Inspiral, and Epigenomics workflow on the Lille and Sophia sites, respectively. For the Montage workflow as shown in Figure 15 and 16, we have seen that the AFSA algorithm has a PSR value close to other algorithms for the Lille and Sophia sites. For the Montage workflow, on the two sites, all the algorithm has a lower PSR when the budget and deadline constraints are tight. For the Inspiral workflow, Figures 17 and 18 depict the PSR values under the Lille and Sophia sites, respectively. DBCS, WMFCO, and AFSA have a better PSR value when the constraints are loose. When the constraints are tight, all the algorithms have a lower PSR. The algorithms have a better PSR on the Sophia site rather than the Lille site. AFSA has a similar PSR compare to WMFCO and DBCS. For the Epigenomics workflow, Figures 19 and 20 show the PSR values under the Lille and Sophia sites. AFSA has a better performance compared to BHEFT, DBCS, and

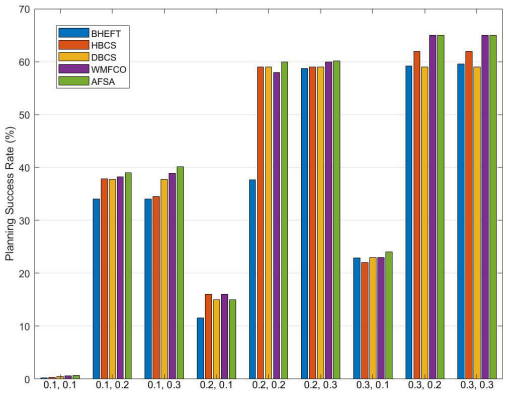

FIGURE 17. The PSR for Inspiral on the Lille site.

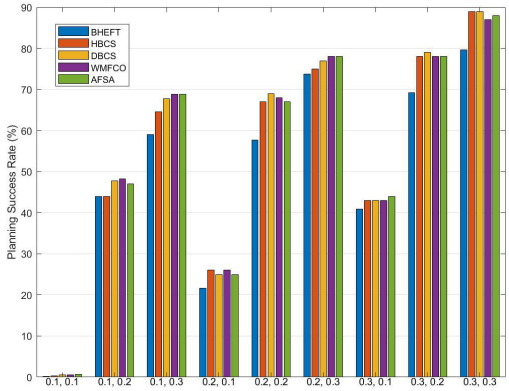

FIGURE 18. The PSR for Inspiral on the Sophia site.

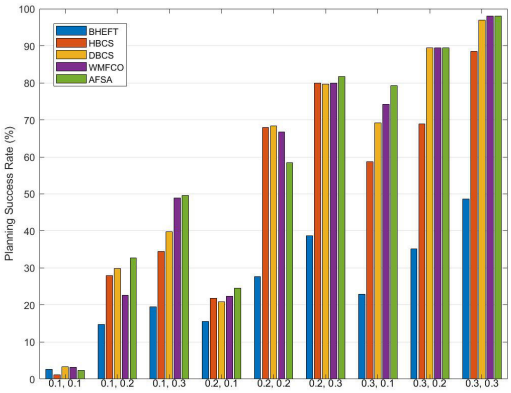

FIGURE 19. The PSR for Epigenomics on the Lille site.

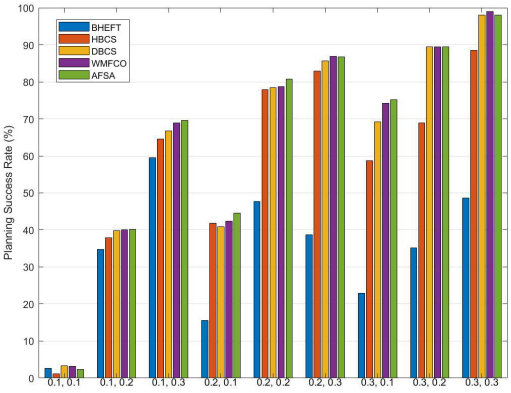

FIGURE 20. The PSR for Epigenomics on the Sophia site.

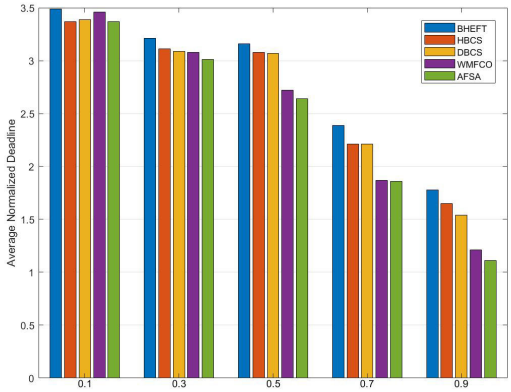

FIGURE 21. Average normalized deadline versus budget constraint for Montage on the Lille site.

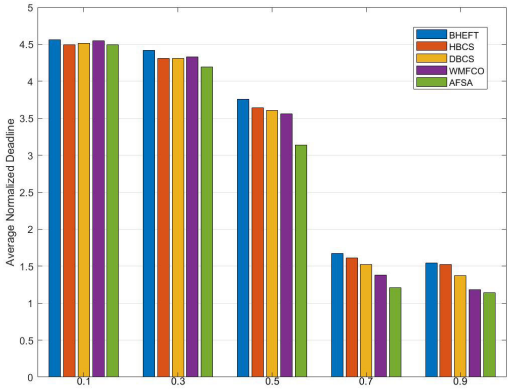

FIGURE 22. Average normalized deadline versus budget constraint for Montage on the Sophia site.

WMFCO when the constraints are loose. The algorithms have a better PSR on the Lille site than the Sophia site when the constraints are loose. By comparing these two figures, we see that HBEFT has a lower PSR than the other algorithms. By comparing other workflows, we see that the Montage workflow with a higher ccr has a lower PSR. This is a result what we expect. As we know from Table 7, the Montage workflow has a lower PSR with the ccr increases. Based on those results, we can evaluate the scheduling performance based on the deadline and cost constraints and workflow type.

We also compare the ND values of the studied five algorithms for the real-world workflow applications. We use the workflow of Montage with 98 tasks, Inspiral with 120 tasks, and Epigenomics with 100 tasks in our evaluation. For each workflow, we repeated 100 times with different ccr values and used these five algorithms for scheduling with different budget parameters and then recorded their average normalized makespan.

Figures 21 and 22 give the average normalized deadline for the Montage workflow with different budget parameters on the Lille and Sophia sites, respectively. By comparing these two figures, we can know that the Sophia site has a larger

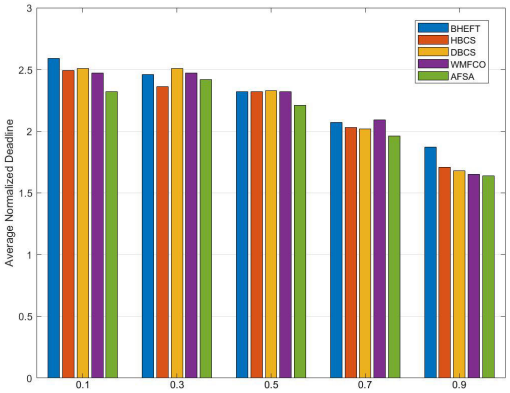

FIGURE 23. Average normalized deadline versus budget constraint for Inspiral on the Lille site.

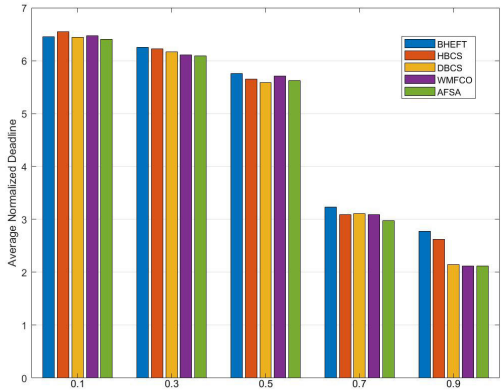

FIGURE 24. Average normalized deadline versus budget constraint for Inspiral on the Sophia site.

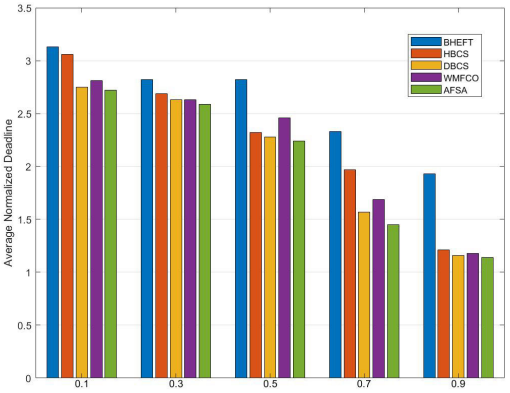

FIGURE 25. Average normalized deadline versus budget constraint for Epigenomics on the Lille site.

ND value than the Lille site, which implies that the Lille site processors have a higher processing capacity than the Sophia site processors since they have the same budget constraint. When $\phi_b = 0.9$, we see that the average normalized deadline of AFSA is 1.11 that is closer to the 1. Figures 23 and 24 show the average normalized deadlines for the Inspiral workflow with 120 tasks for different budget parameters on the Lille and Sophia sites, respectively. From Figure 23, we see that AFSA has the lowest average ND when ϕ_b is more than 0.5.

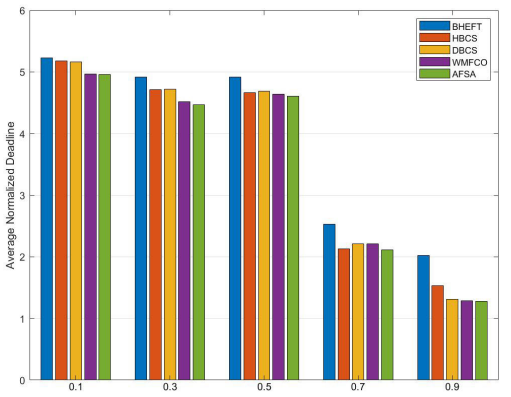

FIGURE 26. Average normalized deadline versus budget constraint for Epigenomics on the Sophia site.

When $\phi_b = 0.1$ and $\phi_b = 0.3$, that is, when the budget is tight, BHEFT, HBCS, DBCS, WMFCO, and AFSA have the same average ND. As shown in Figure 24, the average ND values of these five algorithms is almost identical. By comparing these two figures, we can know that the Sophia site has a higher ND value than the Lille site. Figures 25 and 26 show the average normalized deadline for the Epigenomics workflow with 100 tasks for different budget parameters on the Lille and Sophia sites, respectively. From Figure 25, we see that the BHEFT algorithm has a higher average ND than the other four algorithms. When $\phi_b = 0.9$, the ND values of AFSA and DBCS are closer to 1. As shown in Figure 26, when $\phi_b = 0.9$, that is, when the budget is loose, DBCS, WMFCO, and AFSA have similar average values of ND, and they are closer to 1.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an aggregation measure factor-based scheduling algorithm for science workflow applications. The proposed algorithm considered budget and deadline for workflow processing. In order to test the performance of the AFSA algorithm, we compared our algorithm with the DBCS, HBCS, BHEFT, and WMFCO algorithms. The experiments showed that AFSA has an equal or higher PSR compared with the DBCS, HBCS, and HBEFT algorithms. The AFSA algorithm has a higher PSR compare to the WMFCO algorithm under the tight deadline and budget constraints. To assess the balance of budget and deadline constraints during the scheduling, we introduced a balance factor (BF) to evaluate our proposed algorithm and other existing algorithms. Our experimental results showed that our AFSA algorithm has a better BF compared with the DBCS, HBCS, BHEFT, and WMFCO algorithms.

Future work will consider priority-type workflow application scheduling subject to budget and deadline constraints. Our work will also focus on both scheduling strategy and DVFS together. Furthermore, we plan to consider other parameters such as task utilization in the scheduling problem of workflow applications.

REFERENCES

[1] H. Arabnejad, J. G. Barbosa, and R. Prodan, "Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources," *Future Gener. Comput. Syst.*, vol. 55, pp. 29–40, Feb. 2016.
[2] N. Zhou, F. Li, K. Xu, and D. Qi, "Concurrent workflow budget- and deadline-constrained scheduling in heterogeneous distributed environments," *Soft Comput.*, vol. 22, no. 23, pp. 7705–7718, Dec. 2018.
[3] E. G. Coffman and J. L. Bruno, *Computer and Job-Shop Scheduling Theory*. Hoboken, NJ, USA: Wiley, 1976.
[4] C.-C. Hsu, K.-C. Huang, and F.-J. Wang, "Online scheduling of workflow applications in grid environment," in *Proc. Int. Conf. Grid Pervas. Comput.* Berlin, Germany: Springer, 2010, pp. 300–310.
[5] H. Arabnejad and J. G. Barbosa, "Maximizing the completion rate of concurrent scientific applications under time and budget constraints," *J. Comput. Sci.*, vol. 23, pp. 120–129, Nov. 2017.
[6] H. Arabnejad, J. G. Barbosa, and F. Suter, "Fair resource sharing for dynamic scheduling of workflows on heterogeneous systems," *High-Perform. Comput. Complex Environ.*, vol. 95, pp. 147–167, Jun. 2014.
[7] A. Dogan, "Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems," *Comput. J.*, vol. 48, no. 3, pp. 300–314, Mar. 2005.
[8] R. Prodan and M. Wieczorek, "Bi-criteria scheduling of scientific grid workflows," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 2, pp. 364–376, Apr. 2010.
[9] G. Singh, C. Kesselman, and E. Deelman, "A provisioning model and its comparison with best-effort for performance-cost optimization in grids," in *Proc. 16th Int. Symp. High Perform. Distrib. Comput. (HPDC)*, 2007, pp. 117–126.
[10] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002.
[11] L. F. Bittencourt, R. Sakellariou, and E. R. M. Madeira, "DAG scheduling using a lookahead variant of the heterogeneous earliest finish time algorithm," in *Proc. 18th Euromicro Conf. Parallel, Distrib. Netw. Process.*, Feb. 2010, pp. 27–34.
[12] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682–694, Mar. 2014.
[13] A. K. Maurya and A. K. Tripathi, "On benchmarking task scheduling algorithms for heterogeneous computing systems," *J. Supercomput.*, vol. 74, no. 7, pp. 3039–3070, Jul. 2018.
[14] H. M. Fard, R. Prodan, and T. Fahringer, "A truthful dynamic workflow scheduling mechanism for commercial multicloud environments," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1203–1212, Jun. 2013.
[15] F. Wu, Q. Wu, and Y. Tan, "Workflow scheduling in cloud: A survey," *J. Supercomput.*, vol. 71, no. 9, pp. 3373–3418, Sep. 2015.
[16] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Sci. Program.*, vol. 14, nos. 3–4, pp. 217–230, 2006.
[17] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control," *J. Grid Comput.*, vol. 11, no. 4, pp. 633–651, Dec. 2013.
[18] H. Arabnejad and J. G. Barbosa, "A budget constrained scheduling algorithm for workflow applications," *J. Grid Comput.*, vol. 12, no. 4, pp. 665–679, Dec. 2014.
[19] T. Gao, C. Q. Wu, A. Hou, Y. Wang, R. Li, and M. Xu, "Minimizing financial cost of scientific workflows under deadline constraints in multi-cloud environments," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2019, pp. 114–121.
[20] T. Sun, C. Xiao, and X. Xu, "A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained," *Cluster Comput.*, vol. 22, no. S3, pp. 5987–5996, May 2019.
[21] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," 2009, *arXiv:0901.0131*. [Online]. Available: http://arxiv.org/abs/0901.0131
[22] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, "Scheduling workflows with budget constraints," in *Proc. Integr. Res. GRID Comput.* Boston, MA, USA: Springer, 2007, pp. 189–202.
[23] L. Zeng, B. Veeravalli, and X. Li, "ScaleStar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud," in *Proc. IEEE 26th Int. Conf. Adv. Inf. Netw. Appl.*, Mar. 2012, pp. 534–541.
[24] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in *Proc. Int. Conf. Comput. Intell. Secur.*, Dec. 2010, pp. 184–188.
[25] R. Sakellariou and H. Zhao, "A hybrid heuristic for DAG scheduling on heterogeneous systems," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, 2004, p. 111.
[26] S. K. Garg, R. Buyya, and H. J. Siegel, "Time and cost trade-off management for scheduling parallel applications on utility grids," *Future Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1344–1355, Oct. 2010.
[27] Y. Choon Lee, R. Subrata, and A. Y. Zomaya, "On the performance of a dual-objective optimization model for workflow applications on grid platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 9, pp. 1273–1284, Sep. 2009.
[28] K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," in *Proc. IEEE 5th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 638–645.
[29] A. K. M. K. A. Talukder, M. Kirley, and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global grids," *Concurrency Comput., Pract. Exper.*, vol. 21, no. 13, pp. 1742–1756, Sep. 2009.
[30] Y. Yuan, H. Li, W. Wei, and Z. Lin, "Heuristic scheduling algorithm for cloud workflows with complex structure and deadline constraints," in *Proc. Chin. Control Conf. (CCC)*, Jul. 2019, pp. 2279–2284.
[31] W.-N. Chen and J. Zhang, "An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 39, no. 1, pp. 29–43, Jan. 2009.
[32] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT," *Future Gener. Comput. Syst.*, vol. 93, pp. 278–289, Apr. 2019.
[33] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment," *Simul. Model. Pract. Theory*, vol. 87, pp. 311–326, Sep. 2018.
[34] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, and M. Chen, "Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud," *J. Syst. Archit.*, vol. 84, pp. 12–27, Mar. 2018.
[35] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: A budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1239–1254, Jun. 2020.
[36] A. Ilyushkin, A. Bauer, A. V. Papadopoulos, E. Deelman, and A. Iosup, "Performance-feedback autoscaling with budget constraints for cloud-based workloads of workflows," 2019, *arXiv:1905.10270*. [Online]. Available: http://arxiv.org/abs/1905.10270
[37] N. Anwar and H. Deng, "Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments," *Future Internet*, vol. 10, no. 1, p. 5, 2018.
[38] *DAGGEN*. Accessed: 2013. [Online]. Available: https://github.com/frs69wq/daggen
[39] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, no. 3, pp. 682–692, Mar. 2013.

TING SUN is currently pursuing the Ph.D. degree with the Faculty of Information Technology, Beijing University of Technology. She was a Visiting Ph.D. Student with the University of South Florida from 2017 to 2019. Her research interests are in the areas of resources scheduling and cloud computing.

YAQIN ZHANG is currently pursuing the Ph.D. degree with the Faculty of Information Technology, Beijing University of Technology.

KAIQI XIONG received the Ph.D. degree in computer science from North Carolina State University. He is currently a Professor with the University of South Florida, affiliated with the Florida Center for Cybersecurity, the Department of Mathematics and Statistics, and the Department of Electrical Engineering. Before returning to academia, he had worked in IT industry for several years. His research interest includes security, networking, and data analytics via machine learning, including deep learning with applications cyber-physical systems, cloud computing, sensor networks, and the Internet of Things (IoT). He received the Best Demo Award at the 22nd GENI Engineering Conference (GEC22) and the US Ignite Application Summit with his team in 2015 as well as the Best Paper Award at several conferences such as, the 2018 IEEE Power and Energy Society General Conference, National Science Foundation (NSF), NSF/BBN, Air Force Research Laboratory (AFRL), Amazon AWS, Florida Center for Cybersecurity (FC2), and Office of Naval Research (ONR) have recently supported his research.

CHUANGBAI XIAO received the Ph.D. degree from Tsinghua University, in 1995. Since 2001, he has been teaching and researching with the Faculty of Information Technology, Beijing University of Technology, where he is currently a Professor. He has authored or coauthored over 100 papers in peer-reviewed journals, conferences, or workshops.