IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 6, 2020, accepted January 21, 2020, date of publication January 30, 2020, date of current version February 7, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2970475*

# Scheduling Scientific Workflow Using Multi-Objective Algorithm With Fuzzy Resource Utilization in Multi-Cloud Environment

**MAZEN FARID[1,2], ROHAYA LATIP[1], MASNIDA HUSSIN[1], (Member, IEEE), AND NOR ASILAH WATI ABDUL HAMID[1]**

[1]Department of Communication Technology and Networks, Universiti Putra Malaysia (UPM), Serdang 43400, Malaysia
[2]Faculty of Education-Saber, University of Aden, Aden, Yemen

Corresponding author: Mazen Farid (mazenfareed7@yahoo.com)

This work was supported in part by Putra Grant, University Putra Malaysia, under Grant 95960000, and in part by the Ministry of Education (MOE) Malaysia. Utmost appreciation and thanks to provide sufficient facilities and funding through-out this research.

**ABSTRACT** The provision of resources and services for scientific workflow applications using a multi-cloud architecture and a pay-per-use rule has recently gained popularity within the cloud computing research domain. This is because workflow applications are computation intensive. Most of the existing studies on workflow scheduling in the cloud mainly focus on finding an ideal makespan or cost. Nevertheless, there are other important quality of service metrics that are of critical concern in workflow scheduling such as reliability and resource utilization. In this respect, this paper proposes a new multi-objective scheduling algorithm with Fuzzy resource utilization (FR-MOS) for scheduling scientific workflow based on particle swarm optimization (PSO) method. The algorithm minimizes cost and makespan while considering reliability constraint. The coding scheme jointly considers task execution location and data transportation order. Simulation experiments reveal that FR-MOS outperforms the basic MOS over the PSO algorithm.

**INDEX TERMS** Multi-objective optimization, multi-cloud environment, reliability, particle swarm optimization, workflow scheduling.

## I. INTRODUCTION

Nowadays, cloud computing technology has become one of the most prominent technologies that provide computing resources to end users. An interesting aspect of this paradigm is that the resources provided can be accessed in the form of utility where customers can pay for the services they use [1]–[5]. To run cloud-based applications in a cost-effective and scalable manner, deploying large scale virtual machines (VMs) is a highly auspicious consideration [6], [7].

In scientific computing applications such as astronomy, physics and bioinformatics, a workflow is the most widely used model for representing scheduled tasks [8]–[10]. Such tasks are usually computation-intensive and thus require high-performance computing machines provided by Cloud computing in a distributed manner [11].

An offshoot of the traditional cloud computing model is a multi-cloud environment where various cloud-based Infrastructure as a Service (IaaS) providers (e.g., Microsoft Azure, 2018; Amazon EC2, 2018; Google Compute Engine, 2018) with diverse resource options can cater for their computing needs by making their virtual machines available at a price. This is one of the most promising solutions available for cloud service providers to share their resources [12], [13].

The question of how to schedule workflow in a multi-cloud computing environment is a quite complicated problem [12], [14] which is regarded as NP-complete [15]. This is because independent cloud IaaS offers this service by putting their computing resources together. Particularly, meeting the quality of service requirements is a daunting challenge since selecting the optimal combination of services from these independent IaaS platforms is somewhat difficult [9], [11], [16].

Like other distributed systems, cloud computing is vulnerable to software faults, hardware failures and power malfunction [17]. These unavoidable issues lead to task and workflow failures during the course of executing sophisticated workflow applications [18], [19]. Hence, it is important to ensure reliability while scheduling workflow in clouds

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

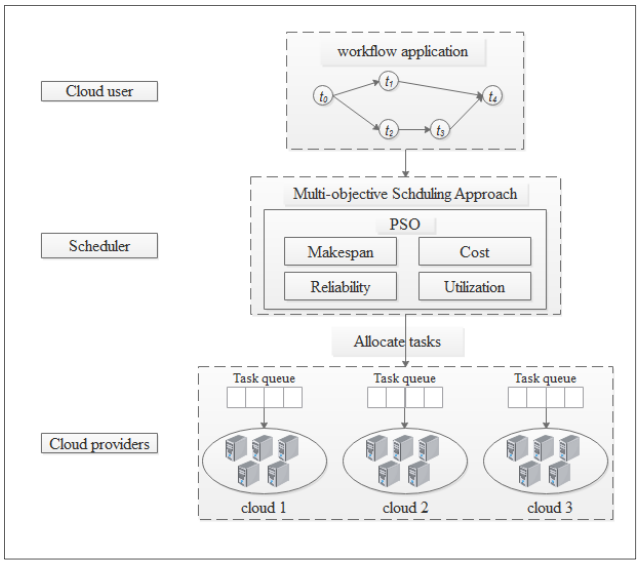



**FIGURE 1.** Scheduling model.

[20]. Although cloud providers consider different reliability parameters, it is important that users pay attention to the workflow's reliability constraints.

In this paper, we propose new multi-objective scheduling (MOS) algorithm with Fuzzy resource utilization (FR-MOS) to schedule scientific workflow in multi-cloud environments. The main aim of the proposed algorithm is to minimize cost and makespan, taking into account reliability constraints. In this regard, the scientific workflow schedule accounts for the following issues: (1) the IaaS cloud platform to be selected; (2) the type of VM to be assigned to the tasks; and (3) the order of tasks that should be utilized for data transmission. To address these issues, the FR-MOS algorithm deploys particle swarm optimization (PSO) and considers task orders and task execution location in its coding strategy. The FR-MOS structure is illustrated in Figure 1. Simulation results show that the FR-MOS algorithm gives better results than MOS algorithm [21] when deployed for scheduling different scientific real-world workflow.

### A. CONTRIBUTIONS

This paper achieves the following key contributions:

(1) Determination of the reliability constraint coefficient $\rho$ in the proposed FR-MOS algorithm using fuzzy logic with resource utilization. This is to minimize cost and makespan under reliability constraint. This procedure was carried out considering three commercial clouds (Amazon EC2, Google Compute Engine and Microsoft Azure) and their VM instances.

(2) Integration of PSO method with FR-MOS algorithm. All the VMs provided by different clouds are designed to constitute the entire search space in which the PSO searches the execution location and the particles move to selected VMs by tasks. Moreover, this process depends on the task execution location and the order of tasks.

(3) Application of the proposed FR-MOS algorithm on four real-life scientific workflows. The results are compared with the MOS algorithm using Q-metrics, FS-metrics and S-metrics to measure the multi-objective performance in terms of convergence, diversity and uniformity of the Pareto front.

The rest of this paper is organized thus: Section 2 presents an overview of the prior art on the performance optimization of workflow for scheduling applications. Section 3 defines the scheduling model and describes the problem formulation and network model. The algorithm implementation is detailed in Section 4. Section 5 discusses the experimental results and Section 6 concludes this paper and suggests future work.

## II. RELATED WORKS

The main objective of scheduling algorithms is to find the best resources in the cloud for the applications (tasks) of the end user. This improves the quality of service (QoS) parameters and resource utilization [22]. Its performance can be measured using many parameters or performance indicators, e.g., makespan, reliability, execution cost, execution time, scalability, power consumption, etc. In cloud computing, we must analyze and optimize these parameters using an effective resource scheduling algorithm to meet the requirements of end users and the service provider without affecting the SLA violation. Resource scheduling becomes an interesting issue in cloud computing due to heterogeneity, dynamics and dispersion of resources that are not resolved by existing scheduling algorithms. Therefore, we need a scheduling algorithm that distributes the heterogeneous workload among cloud resources VMs. The algorithm should be based on resource capacity and overcome the problem of overload and underload.

In this regard, it is important to optimize the performance of scheduling applications' workflow. To achieve this, the pertinent objectives for satisfying users' QoS constraints include reducing execution time and total execution cost. In line with this, A cost-minimizing heuristic algorithm was suggested by [9] to schedule scientific workflow under deadline constraints in a cloud computing environment. In order to put idle resources to effective use, [23] introduced a replication strategy that manages the impact of variable resource performance to meet a flexible workflow deadline. Similarly, [24] developed (ADAS) scheduling algorithm for scheduling large scale workflow. The data-aware algorithm comprises configuration and runtime phases. It schedules cloud resources and the workflow setup process while considering cloud security constraints.

Authors of [8] designed (SCAS) algorithm for scheduling scientific workflow having cost and security constraints. The algorithm pertinently considers applications that are data, memory and computation intensive. A safety plan and budget programming algorithm (SABA) was proposed by [25] to improve the makespan while considering security and budget constraints. Most of the above works focus on only one optimization objective. However, users' requirements are diverse

thus, single objective workflow scheduling algorithms fail to meet their requirements.

One of the most critical concerns in cloud computing is the reliability of workflow scheduling. In a distributed cloud computing environment, [26] outlined some important objectives of the scheduling process for a real-world workflow. The authors focus on four objectives: cost, makespan, reliability and energy. A heuristic method that reduces makespan and improves reliability, while considering the probability of failures that may occur in a multi computing environment, was invented in [17]. Authors in [19] proposed an adaptive 'just-in-time' workflow scheduling algorithm to provide fault tolerance and improve reliability using the task resemblance method.

In the workflow scheduling paradigm, the objectives of minimizing energy and maintaining reliability regularly conflict. To reduce energy consumption and increase the reliability of the system, [27] proposed a bi-objective genetic algorithm (BOGA). Reference [28] used an ant colony-based scheduling technique by adapting three ants to improve workflow reliability under deadlines and budget constraints. In the same line, Kaur and Singh [29] applied the budget constraint to minimize makespan and maximize reliability in a cloud computing environment using their workflow scheduling algorithm.

Garg and Singh developed two improved versions of PSO: 1) The non-dominant PSO [NSPSO] [30] that extends the basic form of PSO by utilizing the best results of personal molecules and their offspring to make effective comparisons of non-dominance. 2) The $\varepsilon$-Fuzzy PSO [31] that basically deploys PSO to better identify risk solutions.

Most of the aforementioned studies on multi-objective cloud computing workflow scheduling consider system reliability. We realize it is practical to use these existing methods in our proposed case study, therefore, we consider multi-objective scheduling in terms of makespan, cost, reliability and resource utilization.

## III. SCHEDULING MODEL

The proposed FR-MOS algorithm is designed to consider four QoS requirements, namely makespan, cost, reliability and resource utilization. Figure 1. shows the scheduling model. The first step involves creating a workflow application by cloud users using any structure. The workflows are assigned to a suitable cloud platform with the VM types that fulfil the workflow requirements. Each cloud provider has a task queue and these tasks are processed based on the structure of the workflow. Since cloud users can access an infinite amount of VM resources, concurrent services can be delivered for tasks in parallel queues while sequencing tasks must be performed based on dependency relationships. Note that each cloud service provider in the considered multi-cloud environment has its own performance metrics and pricing model.

This section discusses the problem of workflow scheduling in detail. Particular attention is given to the following: workflow model, multi-cloud model, computation of makespan, cost reliability and resource utilization, and problem formulation. Table 1 defines the notations used in this study.

**TABLE 1. Notations and their meanings.**

| Notation | Meaning |
|---|---|
| $t_i$ | Task $t_i$ of workflow |
| $D(t_j, t_i)$ | Size of data transmission from task $t_i$ to $t_j$ |
| $W(t_i)$ | Workload of task $t_i$ |
| $pre(t_i)$ | Predecessor set of $t_i$ |
| $succ(t_i)$ | Successor set of $t_i$ |
| $n$ | Number of workflow tasks |
| $VM(m)$ | VM type set of cloud IaaS platform $m$ |
| $VM(m,k)$ | Type of VM $k$ in IaaS platform $m$ |
| $P(m,k)$ | Processing capacity of $VM(m,k)$ |
| $c(m,k)$ | Cost per unit time of $VM(m,k)$ |
| $B_m$ | Bandwidth of cloud IaaS platform $m$ |
| $B_{mm'}$ | Bandwidth between platform $m$ and $m'$ |
| $T_{trans}(t_i, t_j)$ | Transmission time from $t_i$ to $t_j$ |
| $T_{exec}(t_i, VM(m,k))$ | Execution time of $t_i$ on $VM(m,k)$ |
| $T_{start}(t_i)$ | Start time of task $t_i$ |
| $T_{end}(t_i)$ | End time of task $t_i$ |
| $T_{rent}(t_i, VM(m,k))$ | VM rent time of task $t_i$ on $VM(m,k)$ |
| $cost(t_i, VM(m,k))$ | VM rent cost of task $t_i$ on $VM(m,k)$ |
| $rel(t_i)$ | Reliability of task $t_i$ |
| $cost$ | Cost of workflow |
| $makespan$ | Makespan of workflow |
| $reliability$ | Reliability of workflow |
| $rel_c$ | Reliability constraint of workflow |
| $VMs_{requestedMIPS}$ | Total processing capacity of requested $VM(m,k)$ |
| $VMs_{availableMIPS}$ | Total available processing capacity |
| $utilization$ | Total resource utilization of workflow |

### A. WORKFLOW MODEL

To achieve a stated objective in any environment, it is necessary to have a sequence of planned activities. Such activities are referred to as workflows. Workflows involve collections of simple processes that are targeted at solving a more complex problem [32]. To ensure efficiency and improve the execution of desired tasks, these processes must follow a particular pattern. Hence, the aim of a workflow is to define how various tasks are configured, performed and tracked.

Workflow can be modeled as a Direct Acyclic Graph (DAG) having nodes and edges. It can be represented as $W = (T, E)$, where $T = \{t_0, t_1, \ldots, t_i, \ldots, t_{n-1}\}$ is the set of tasks. Task dependencies are represented by a set of arcs, $E = \{(t_i, t_j)|t_i, t_j \epsilon T\}$. Each workflow has an entry task $T_{entry}$ and exit task $T_{exit}$. Also, each task has predecessor set denoted by $pre(t_i)$ and successor set denoted by $succ(t_i)$. The task is executed after its predecessor's set execution. Task $t_i$ has an assigned weight $W(t_i)$ which denotes its workload and is quantified in terms of compute unit (CU). The data output size of tasks $t_i$ needed to be transferred to task $t_j$ is represented by $D(t_i, t_j)$.

### B. MULTI-CLOUD ARCHITECTURE

The studied multi-cloud environment in this paper allows users to access VMs of different cloud providers with a

**TABLE 2.** Different types of VMs in Amazon EC2.

| VM type | VM type no. | Compute unit | Cost per hour ($) |
|---|---|---|---|
| m1.small | 1 | 1.7 | 0.06 |
| m1.medium | 2 | 3.75 | 0.12 |
| m1.large | 3 | 7.5 | 0.24 |
| m1.xlarge | 4 | 15 | 0.45 |
| m3.2xlarge | 5 | 30 | 0.9 |

**TABLE 3.** Different types of VMs in google compute engine.

| VM type | VM type no. | Compute unit | Cost per hour ($) |
|---|---|---|---|
| n1-highcpu-2 | 1 | 2 | 0.07 |
| n1-highcpu-4 | 2 | 4 | 0.14 |
| n1-highcpu-8 | 3 | 8 | 0.28 |
| n1-highcpu-16 | 4 | 16 | 0.56 |
| n1-highcpu-32 | 5 | 32 | 1.12 |

**TABLE 4.** Different types of VMs in microsoft azure.

| VM type | VM type no. | Compute unit | Cost per hour ($) |
|---|---|---|---|
| B2S | 1 | 1 | 0.0496 |
| B2MS | 2 | 2 | 0.0912 |
| B4MS | 3 | 4 | 0.182 |
| B8MS | 4 | 8 | 0.365 |
| B16MS | 5 | 16 | 0.73 |

variety of price mechanisms. For example, in (Amazon EC2, 2018), rates are based on working hours. This implies that each partial hour finally reaches full hours. Similarly, (Microsoft Azure, 2018) charges customers per minute. Unlike others, (Google Compute Engine, 2018) charges the first ten minutes then the VM instances are later loaded in minutes. A multi-cloud environment can provide different numbers of $m$ IaaS platforms with each supplying a set of VMs, where $VM(m) = \{VM(m,1), \ldots, VM(m,k), \ldots, VM(m,k_s)\}$, $m = 1, 2, \ldots, M$.

This paper considers three cloud service providers: Amazon EC2, Google Compute Engine, Microsoft Azure. Tables 2-4 show the different types of VMs. Let $(m|m = 1, 2, \text{and } 3)$ where $m$ represents the different providers of IaaS cloud (i.e., Amazon, Microsoft and Google). $VM(m,k)$ is the instance of the VM specified by the IaaS cloud provider $m$ with CU processing capacity $P(m,k)$ and cost per hour $C(m,k)$, where CU represents the CPU computing capacities of VMs [16], [33]. We assume that the different cloud providers can provide an infinite number of VMs to the end users. $B_m$ indicates the bandwidth of $m$ cloud platform and $B_{mm'}$ is the bandwidth between two different cloud platforms $m$ and $m'$.

### C. MAKESPAN COMPUTATION

Workflow tasks can be assigned and executed on different IaaS platforms in a multi-cloud environment. Most of the previous studies compute the start time of the current task as the latest finish time of the previous task. Therefore, some virtual machines must wait to receive tasks when it is their turn. This is because VMs send multiple copies of their output to other VMs. The layout of the recipient output depends on the order of the task. Given a set A, defined in Eq. (1), the order of tasks in the workflow is taken into account to sort tasks in the group.

$$A = \sum_{t_j \in pre(t_i)} succ\left(t_j\right). \tag{1}$$

The order of tasks in partial set B follows the same sequence as the previous tasks $t_i$ in set A, if $A = \{t_1, t_3, t_4, t_2\}$, then $B = \{t_1, t_3\}$ when $t_i = t_4$. The start time of the task $t_i$ is represented as $T_{start}(t_i)$, and the end time is represented as $T_{end}(t_i)$, Eq. (2) represents the start time of $t_i$.

$$T_{start}\left(t_i\right) = max_{t_j \in pre(t_i)} \left\{T_{end}\left(t_j\right) + T_{wait}\left(t_j, t_i\right)\right\}, \tag{2}$$

where $T_{wait}(t_j, t_i)$ is the waiting time of task $t_i$ to receive input data from task $t_j$. It is expressed thus:

$$T_{wait}\left(t_j, t_i\right) = \sum_{t_z \in B} T_{trans}\left(t_j, t_z\right). \tag{3}$$

Note that if $t_i = t_{entry}$, then $T_{start}(t_i) = 0$.
To compute the finish transmission time, we have

$$T_{trans}\left(t_i\right) = max_{t_j \in pre(t_i)} \left\{T_{end}\left(t_j\right) + T_{wait}\left(t_j, t_i\right)\right\} + T_{trans}\left(t_j, t_i\right), \tag{4}$$

where $T_{trans}(t_j, t_i)$ is the transmission time between $t_i$ and $t_j$. Notwithstanding, two cases (i.e., $t_i$ and $t_j$) should be considered. These are $T_{trans}$ and $T_{rece}$.

$$T_{trans} = \begin{cases} D\left(t_j, t_i\right)/B_m, \\ D(t_j, t_i)/B_{mm'}, \quad m \neq m' \end{cases} \tag{5}$$

Hence, task $t_i$ has a receiving time given by

$$T_{rece}\left(t_i\right) = T_{trans}\left(t_i\right) - T_{start}\left(t_i\right). \tag{6}$$

The execution time of each task $t_i$ depends on the output data size of every task [9], [16]. The execution time of different tasks on different $VM(m,k)$ can be calculated by the following equation.

$$T_{exec}\left(t_i, VM\left(m,k\right)\right) = \frac{W\left(t_i\right)}{P\left(m,k\right)}. \tag{7}$$

The end time of each task can be calculated using the processing capacity of $VM(m,k)$ in CU. Thus,

$$T_{end}\left(t_i\right) = T_{start}\left(t_i\right) + T_{rece}\left(t_i\right) + T_{exec}\left(t_i, VM\left(m,k\right)\right). \tag{8}$$

Using the analysis above, we can compute the makespan of the workflow as shown in Eq. (9).

$$makespan = T_{end}\left(t_{exit}\right). \tag{9}$$

The makespan equates the end time of the last task $(t_{exit})$.

### D. COST COMPUTATION

Following the profit-making multi-cloud model, IaaS platforms have unique pricing methods. The existing workflow algorithms calculate the VM rent time by taking the interval between the task execution start time and end time [9], [11], [16]. When is task is completed, the VM shuts down and the output of that task is transferred to its successors. The priority of data transfer depends on the tasks' order. Then the send time of task $t_i$ can be expressed as:

$$T_{send}(t_i) = \sum_{t_j \in succ(t_i)} T_{trans}(t_i, t_j). \tag{10}$$

The rent time of task $t_i$ for the VM which is run on $VM(m,k)$ is given in Eq. (11).

$$T_{rent}(t_i, VM(m,k)) = T_{rece}(t_i) + T_{exec}(t_i, VM(m,k)) + T_{send}(t_i). \tag{11}$$

VM rent cost of task $t_i$ for each considered IaaS platform is calculated below.

For Amazon EC2 that charges per hour, the execution cost of task $t_i$ on $VM(1,k)$ is expressed in Eq. (12).

$$cost(t_i, VM(1,k)) = \lceil T_{rent}(t_i, VM(1,k))/T_{minute} \rceil \cdot C(1,k), \tag{12}$$

where $T_{minute} = 60$.

Microsoft Azure charges per minute, the execution cost of task $t_i$ on $VM(2,k)$ is expressed in Eq. (13).

$$cost(t_i, VM(2,K)) = T_{rent}(t_i, VM(2,k)).C(2,k)/T_{minute}. \tag{13}$$

After the first ten minutes, Google charges the VM instance per minute, the execution cost of task $t_i$ on $VM(3,k)$ is shown in Eq. (14), where $T_{ten} = 10$.

$$cost(t_i, VM(3,k)) = \begin{cases} T_{ten}.C(3,k)/T_{minute}, & if\ T_{rent}(t_i, VM(3,K)) \le T_{ten} \\ T_{rent}(t_i, VM(3,k)).C(3,k)/T_{minute}, & otherwise \end{cases} \tag{14}$$

The workflow cost can be calculated using Eq. (15).

$$cost = \sum_{t_i \in T} cost(t_i, VM(m,k)). \tag{15}$$

### E. RESOURCE UTILIZATION COMPUTATION

In cloud computing, scheduling plays an important role of efficiently allocating resources to each task. Most scheduling processes are accomplished by allocating tasks to balance between improved performance in terms of makespan, cost and efficient resource utilization [34]. Cheaper resources take more time than expensive resources. This implies that the high processing capacities of CPUs in various VMs improves their performances while delivering better makespan-cost trade-offs. The total processing capacity of a requested VM is calculated thus:

$$VMs_{requestedMIPS} = \sum P(m,k). \tag{16}$$

And the percentage of workflow resource utilization of each workflow can be computed using Eq. (17).

$$utilization = \frac{VMs_{requestedMIPS}}{VMs_{availableMIPS}} * 100. \tag{17}$$

### F. RELIABILITY COMPUTATION

Failures are unavoidable in a cloud computing environment. Also, faults (such as software faults, hardware failures and power malfunction) can come from the inside [19], [35] and outside (e.g., harmful attacks from the internet) [8], [25]. Short term faults happen frequently and cause failure during the execution of workflow tasks. The occurrence of failure can be modelled as a Poisson distribution [17], [27], [36]. The probability that the task $t_i$ will be performed correctly in $VM(m,k)$ is calculated using the exponential distribution as follows:

$$rel(t_i) = \exp(-\lambda_m.T_{rent}(t_i, VM(m,k))), \tag{18}$$

where the failure coefficient of cloud service provider $\lambda_m > 0$ $(m = 1, 2, 3)$.

In addition, each IaaS platform has a unique failure coefficient in the multi-cloud environment. The execution of the task will fail if any problem occurs during the rental time. The reliability of workflow is calculated using Eq.(19), assuming the failures are independent.

$$reliability = \prod_{t_i \in T} rel(t_i). \tag{19}$$

Suppose $\lambda_{max} = max\{\lambda_m | m = 1,2,3\}$ indicates the maximum failure coefficient and $\lambda_{min} = min\{\lambda_m | m = 1,2,3\}$ indicates the minimum failure coefficient, the resultant workflow for scheduling the reliability can be maximum $(rel_{max})$ or minimum $(rel_{min})$. Moreover, different workflow reliability results are produced according to the scheduling process of tasks in different clouds. Thus, the cloud users have to set an appropriate reliability constraint $rel_c$ for the scientific workflow application, i.e.,$rel_c \epsilon [rel_{min}, rel_{max}]$.

### G. FUZZY LOGIC

In 1965 Lutfy Zadeh presented the concept of fuzzy logic [37]. The theory involves a new mathematical model used to formalize and analyze the features of sets. The fuzzy logic is a natural extension of the agreed common language and the interpretation of human behaviour [38].

*Definition 1:* Let X be an associate absolute reference set. The characteristic performance of every traditional subset of X and A, $\mu_A$: X $\rightarrow$ {0,1} is defined as follows:

$$\mu_A(x) = \begin{cases} 1: & x \in A \\ 0: & x \notin A \end{cases}$$

Depending on definition 1, each x $\in$ X, $\mu_{A(x)}$ will take only one value of the set 0 and 1.

*Definition 2:* If the set $\mu_A$ involves two numbers {0,1}, set to the interval between [1,0], a function is obtained for each member of X which is assigned to a number between the range of [0,1]. Therefore, A is a fuzzy set.

According to Definition 2, assuming $\mu_A(x) \in \{0, 1\}$, then the membership of set A becomes uncertain. Thus, we propose a multi-objective algorithm for scheduling scientific workflow using a fuzzy system working with the resource utilization to determine the reliability constraint coefficient.

### H. PROBLEM DESCRIPTION

The description above introduced the relationship among performance metrics such as cost, makespan, resource utilization and reliability. The focus of the multi-objective scheduling problem addressed in this paper is on minimizing makespan and cost, and maximizing resource utilization with reliability constraint. The workflow is represented as $WF = (T, E)$. One or more solutions are generated as the primary objective of scheduling $\Gamma = (Loc, Ord, R)$, where $Loc = \{loc(t_0), loc(_{t1}), \ldots, loc(t_{n-1})\}$ is the location of the workflow task that would be executed. $Ord = \{ord(t_0), ord(t_1), \ldots, ord(t_{n-1})\}$ is the order of data transfer tasks that are primarily used to determine the tasks' wait time for the tasks. Additionally, the tasks in $Ord$ must also represent the dependency relationships. $R = \{R_0, R_1, \ldots, R_i, \ldots, R_{n-1}\}$ is the information set of the resources considered for the total workflow, where $R_i = (t_i, VM(m; k), T_{start}(t_i), T_{end}(t_i))$. We formally define the multi-objective optimization problem as shown below.

$$\text{Minimize: } F(\Gamma) = (makespan, cost). \quad (20)$$

$$\text{Maximize: } F'(\Gamma) = (resource\ utilization). \quad (21)$$

$$\text{Subject to: } reliability \geq rel_c. \quad (22)$$

Some previous studies consider the location of the executed task when designing scheduling methods [8], [9], [16]. However, the transmission order of data is disregarded, i.e., its priority is not considered. This is very crucial in the course of designing the scheduling strategy. Besides, when there are different task orders with the same execution location, the scheduling results would be different. For this reason, we consider the priority by which data is transmitted.

## IV. THE PROPOSED ALGORITHM

Multi-objective optimization involves optimizing multi conflicting objectives simultaneously. It is expressed mathematically in Eq. (23).

$$\text{Minimize }: F\left(\vec{x}\right) = \left(f_1\left(\vec{x}\right), f_2\left(\vec{x}\right), \ldots, f_k\left(\vec{x}\right)\right), \quad (23)$$

where $k$ represents the number of objectives, $\vec{x} \in X$ is the decision variables' vector and $X$ is the decision space. Usually, several solutions can be derived using Multi-Objective-Optimization but researchers consider Pareto dominance to compare obtained solutions.
For $\vec{x}_1, \vec{x}_2 \in X$, $\vec{x}_1$ is said to dominate $\vec{x}_2$ if and only if

$$\forall i : f_i\left(\vec{x}_1\right) \leq f_i\left(\vec{x}_2\right) \wedge \exists j : f_j\left(\vec{x}_1\right) < f_j\left(\vec{x}_2\right). \quad (24)$$

In this paper, a solution $\vec{x}^*$ is Pareto optimal if no other solution dominates it. The Pareto front is the set of optimal solutions within the objective space. For the multi-cloud workflow scheduling problem considered in this paper, the $\Gamma*$ schedule dominates $\Gamma$ if both cost and makespan are less than that in $\Gamma$ or at least, one of them is less. The workflow scheduling problem is an NP-complete problem, therefore, it is recommended to use a convergent approach for suboptimal solutions. Nevertheless, an approach that has been proven successful for solving such problems is Evolutionary Algorithms (EA) [16], [39]–[45].

The FR-MOS algorithm proposed in this paper can yield solutions that show different cost-makespan trade-offs of which cloud users can select from. FR-MOS algorithm uses PSO (which is also from the class of EAs) to solve the problem of multi-objective and multi-cloud workflow scheduling. Considering the unique properties of multi-cloud scheduling, current coding strategies cannot be applied directly in our case. Therefore, we compare our proposed algorithm that is based on PSO with the method developed by [21] in a multi-cloud environment. Moreover, the authors in [21] compared the MOS algorithm with two other algorithms CMOHEFT and RANDOM. They showed that MOS algorithm outperformed CMOHEFT and RANDOM algorithms. Based on this premise, we re-implemented only MOS algorithm for comparison since it is the best of the studied algorithms in [21], the benchmark.

### A. PARTICLE SWARM OPTIMIZATION (PSO)

PSO was developed by Kennedy and Eberhart in the year 1995. It was developed as a swarm intelligence algorithm classed as an evolutionary computational method [46]. PSO is quite simple and efficient. Moreover, it simulates the hunting behavior of birds which makes it attract a lot of attention. Initially, the PSO algorithm was used to find the solution for single-objective optimization problems. Its great search potential and capacity led to its expansion for solving multi-objective problems [39], [43], [47]. The basic component of PSO is the particle that moves through the search space. The direction and velocity are used to determine the motion of the particle. The velocity is derived by considering a combination of the best historical positions with random disturbances. Its velocity and position update functions are provided in Equations (25) and (26), respectively.

$$\vec{v}_i \leftarrow w.\vec{v}_i + \varphi_1.rd_1.\left(\vec{p}_i - \vec{x}_i\right) + \varphi_2.rd_2.\left(\vec{g}_i - \vec{x}_i\right), \quad (25)$$

$$\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i, \quad (26)$$

where $w$ is the weight of inertia, $\varphi_1$ and $\varphi_2$ are positive integers and $rd_1$, $rd_2 \in [0, 1]$ are random numbers generated by uniform distribution [48]. Each particle can be represented by a three-dimensional vector: the best local position $\vec{p}_i$, the current position $\vec{x}_i$ and its velocity $\vec{v}_i$. Position $\vec{x}_i$ indicates a filter solution which is determined by the PSO algorithm. Whenever the fitness value for the current position $\vec{x}_i$ is better than the fitness value of the previous position, the current position is stored in the vector $\vec{p}_i$. Finally, the best

global position $\vec{g}_i$ for all particles is determined through the particles' communication [49].

The proposed FR-MOS algorithm for the multi-cloud environment (described in Algorithm 1) is based on PSO. The parameters (for scheduling and PSO) are first initialized (lines 1-6). Thereafter, the performance parameters are evaluated during the workflow scheduling process (lines 17-26). Then, the method to select the feasible solution under two reliability constraints is applied [41] as follows: 1) For possible solutions, the fittest solution is the best (lines 28-31). 2) If all solutions are not feasible, the solution with the least violation of the constraint will be chosen as the best solution [39]. Therefore, only feasible solutions will be stored (lines 35-40). For more than one optimal solution resulting from a multi-objective problem, the selection method is used to determine the optimal position (line 43). The algorithm continues until it meets the end condition (line 7).

### B. CODING STRATEGY

In this study, tasks assigned to optimal locations to execute data transmissions are arranged based on the order of each task. As earlier mentioned, this is to address MOS problems. The coding strategy deployed in this paper is given in Eq. (27).

$$\Gamma_c = (loc\,(t_0)\,, loc\,(t_1)\,, \ldots, loc\,(t_{n-1})\,, \\ ord\,(t_0)\,, ord\,(t_1)\,, \ldots, ord\,(t_{n-1})). \quad (27)$$

The number of parameters in $\Gamma_c$ denotes the dimension of a particle, i.e., $\Omega = 2{\cdot}n$ (see Eq. (27)). The positions from 0 to $n-1$ determine the types of VMs allocated for the tasks. Table 5 shows the search space provided by the three IaaS platforms considered in this study using different types of VMs for all tasks. The parameter $loc(t_i)$ of each task $t_i$ considers two types of information: the specific type of VM and the execution location. The $ord(t_i)$ is the task order coordinate which affects the task's waiting time. Figure 2 shows the encoding plan of the workflow.

In Algorithm 2, the dependencies between tasks must be followed by the order of tasks, i.e., task $t_1$ *is* executed before task $t_2$ if $t_1$ is the predecessor of task $t_2$ and $t_2$ follows $t_1$ in string *Ord*. The set of schedulable tasks is first initialized, i.e., $\alpha = \{t_0\}$ (line 2). In (line 3) the two sets of scheduled tasks $\gamma$ and waiting tasks $\beta$ are set to empty. We can see here that the location in search space is recorded by the flag (line 4). Then set $space = [0, 0]$ to indicate the entry task position that cannot be changed (line 5). The proper solution is selected by using the Euclidean distance (lines 11-15) to remove the selected position from the search space (line 16). In the end, all tasks will be tested and attached to a schedulable set $\alpha$(lines $20-24$). Sequentially, the search space is updated (line 26) to ensure a unique search space for each task in the existing schedulable task set $\alpha$. Our proposed algorithm yields a better trade-off compared to MOS, which is considered as a novel method that deploys the PSO coding plan to manage workflow. Algorithms 1 and 2

**Algorithm 1** FR-MOS

**BEGIN**

1. Set the number of particles *Np*
2. Set $A = \emptyset$; //
   *initially empty archive, record non dominated solution*
3. initialize $\{\vec{v}_i, \vec{x}_i, \vec{p}_i, \vec{g}_i\}_{i=1}^{N}$;
   *// random location and velocity*
4. initialize {*reliability*= *makespan* =*cost*= *utilization* = 0};
5. Set $\{\vec{p}_i = \vec{x}_i, \vec{g}_i =, \vec{x}_i\}_{i=1}^{N}$;
6. calculate $\{p_i, g_i\}_{i=1}^{N}$;
7. **While** $idx < N_{IT}$ *//$N_{IT}$ is the number of iteration time*
8. **for** each particle $i$ to $N_P$
9. $\vec{v}_i \leftarrow w.\vec{v}_i + \varphi_1.rd_1.(\vec{p}_i - \vec{x}_i) + \varphi_2.rd_2.(\vec{g}_i - \vec{x}_i)$; *//update velocity*
10. $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$; *// update position*
11. **for task** $t_i$ in *Ord //traverse tasks in order*
12. **if** $t_i = t_0$*//entry task*
13. Set $T_{start}\,(t_i) = 0$; *// this is also the start time of workflow*
14. **else**
15. Set $T_{start}\,(t_i)$ according to Eq. (2) ;
16. **end if**
17. Compute $T_{rece}\,(t_i)$ based on Eq. (6) ;
18. Compute $T_{exec}\,(t_i)$ based on Eq. (7) ;
19. Compute $T_{end}\,(t_i)$ based on Eq. (8) ;
20. Compute the *rel* $(t_i)$ *b*ased on Eq. (18) ;
21. **end for**
22. Calculate *makespan* according to Eq. (9) ;
23. Calculate *cost* according to Eq. (15) ;
24. Calculate *resource utilization* according to Eq. (17) ;
25. Set *reliability coefficient* $\rho$ according to Eq.(34);
26. Calculate *reliability* according to Eq. (19) ;
27. Define $\theta(\vec{x}_i) = \max(0, rel_c - reliability)$
28. **If** $\theta\,(\vec{x}_i) == 0 \wedge \theta\,(\vec{p}_i) == 0$*// $\vec{x}_i$ and $\vec{p}_i$ are all feasible solutions*
29. **If** $\vec{x}_i \preccurlyeq \vec{p}_i \vee (\vec{x}_i \not\prec \vec{p}_i \wedge \vec{p}_i \not\prec \vec{x}_i)$*// update personal $\vec{p}_i$*
30. Set $\vec{p}_i = \vec{x}_i$;
31. **end if**
32. **else**
33. Set $\vec{p}_i = \vec{x}' = argmin\,\{\theta\,(\vec{x}_i)\,, \theta\,(\vec{p}_i)\}$ ;
34. **end if**
35. **If** $\theta\,(\vec{x}_i) == 0$*// only the feasible solution will be added to A*
36. **for** $\forall \vec{x} \in A \wedge \vec{x}_i \not\prec \vec{x}$*// update A*
37. $A = \{\vec{x} \in A | \vec{x} \not\prec \vec{x}_i\}$; *// remove points dominated by $\vec{x}_i$*
38. $A = A \cup \vec{x}_i$; *//add $\vec{x}_i$ to A*
39. **end for**
40. **end if**
41. $idx++$
42. **end for**
43. Randomly select global optimal psition $\vec{g}_i$;
44. **end while**

**END**

**TABLE 5.** Different VM types of search space.

| Number | VM location | VM type no. | Compute unit | Cost per Hour($) |
|---|---|---|---|---|
| **1** | Cloud (m = 1) | 1 | 1.7 | 0.06 |
| **2** | Cloud (m = 1) | 2 | 3.75 | 0.12 |
| **3** | Cloud (m = 1) | 3 | 7.5 | 0.24 |
| **4** | Cloud (m = 1) | 4 | 15 | 0.45 |
| **5** | Cloud (m = 1) | 5 | 30 | 0.9 |
| **6** | Cloud (m = 2) | 1 | 2 | 0.07 |
| **7** | Cloud (m = 2) | 2 | 4 | 0.14 |
| **8** | Cloud (m = 2) | 3 | 8 | 0.28 |
| **9** | Cloud (m = 2) | 4 | 16 | 0.56 |
| **10** | Cloud (m = 2) | 5 | 32 | 1.12 |
| **11** | Cloud (m = 3) | 1 | 1 | 0.0496 |
| **12** | Cloud (m = 3) | 2 | 2 | 0.0912 |
| **13** | Cloud (m = 3) | 3 | 4 | 0.182 |
| **14** | Cloud (m = 3) | 4 | 8 | 0.365 |
| **15** | Cloud (m = 3) | 5 | 16 | 0.73 |

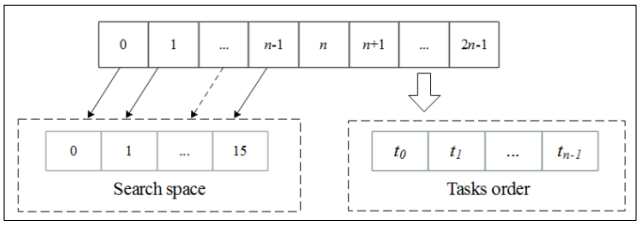



**FIGURE 2.** Encoding approach of workflow.

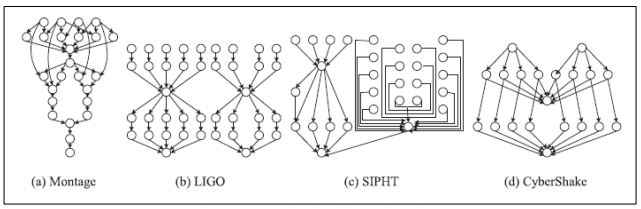



**FIGURE 3.** Structure of scientific workflows [21].

must be integrated to obtain near-optimal multi-objective solutions.

### C. SCHEDULING GENERATION

In the FR-MOS algorithm, the scheduling strategy involves computing the fitness value for each particle, which is used to convert the position of particle $\Gamma_c$ into a workflow $\Gamma$. The start time must be calculated to find the makespan for each task (lines 12-16). Next, receiving data time $T_{rece}(t_i)$, task execution time $T_{exec}(t_i)$ and end time $T_{end}(t_i)$ are calculated (lines 17-19). Then, the reliability of the task is found (line 20). Lastly, the cost, makespan, resource utilization and reliability of workflow scheduling are calculated (lines 22-26).

### D. PERFORMANCE MEASUREMENT

Multipurpose solutions are unlikely to be evaluated on only one of the performance aspects. Here, we use three metrics: Q-metric, S-metric and FS-metric. The essence of using these

**Algorithm 2** Order tasks

**BEGIN**
1. **Initialize**
2. $\alpha = \{t_i\}$ ; *//schedulable entry task* $t_0$
3. $\gamma = \beta = \emptyset$; *// the set of scheduled tasks and temporary tasks*
4. $flag = 0$; *//record location in search space*
5. $space = [0, 0]$ *//search space*
6. **end Initialize**
7. **while** $\alpha \neq \emptyset$
8. $flag = flag + |\alpha|$ ;
9. **for** $t_i$ in $\alpha$
10. Put all succers of task $t_i$ into $\beta$;
11. $\vec{v}_i \leftarrow w.\vec{v}_i + \varphi_1.rd_1.(\vec{p}_i - \vec{x}_i) + \varphi_2.rd_2.(\vec{g}_i - \vec{x}_i)$ ;
12. $\vec{x}_i \leftarrow \vec{x}_i + \vec{v}_i$;
13. **if** $x_i \notin space$
14. $x_i = (x'_i : \min(|x'_i - x_i|, x_i \in space)$
15. **end if**
16. $space = space - \{x_i\}$ ;
17. $\gamma = \gamma + \{t_i\}$ ; *// add task to the set of scheduled tasks*
18. $\alpha = \alpha - \{t_i\}$ ; *//remove task from* $\alpha$
19. **end for**
20. **for** $t_i$ *in* $\beta$
21. **if** $pre(t_i) \in \gamma$
22. $\alpha = \alpha + \{t_i\}$ ; *//add new task to* $\alpha$
23. **end if**
24. **end for**
25. Clear$\beta$;
26. $space = [flag, flag + |\alpha| - 1]$ ; *// update the search space*
27. **end while**

**END**

metrics is to measure the quality of the Pareto fronts found by different algorithms [43]. In particular, to assess the level of convergence of multi-objective algorithms A and B, Q-metric can be deployed [50] [51] as shown in Eq. (28).

$$Q(A, B) = |\Psi| / |Y| . \tag{28}$$

$\Psi = \Upsilon \cap$ SA and Y is the set of SA $\cup$ SB. SA and SB denote two sets of Pareto optimal results for two multi-objective algorithms A and B. Algorithm A has a better performance compared to Algorithm B if and only if Q(A,B) > Q(B, A) or Q(A,B) > 0.5. FS metric is used to determine the Pareto front space size. This is done by using an algorithm in [48]. It is computed in Eq. (29) [31].

$$FS = \sqrt{\sum_{i=1}^{m} \min_{(x_0, x_1) \epsilon SA \times AS} (f_i(x_0) - f_i(x_1))^2}, \tag{29}$$

where $f_i(x_0)$ and $f_i(x_1)$ are two values of one objective function. A larger value of FS means that the Pareto front has

better diversity. To determine the level of uniformity of the solutions, we adopt the S-metric as computed in Eq. (28).

$$s = \sqrt{\sum_{i=1}^{N_P} (d_i' - \bar{d}')^2 \Big/ N_P}, \tag{30}$$

where the number of Pareto solutions is $N_P$ and $d_i'$ calculates the distance between the members of Pareto front set.

$$\bar{d}' = (\sum_{i=1}^{N_P} d_i') \Big/ N_P. \tag{31}$$

As opposed to FS-metric and Q-metric where a larger value is preferred, a smaller S-metric implies the algorithm has discovered a more uniform solution.

## V. PERFORMANCE EVALUATION

In this section, we perform several experiments to evaluate the time complexity of the proposed FR-MOS algorithm. The simulation results are discussed using a real-world workflow and constraints.

### A. EXPERIMENTAL SETUP

The FR-MOS algorithm was implemented on Workflowsim 1.0 using i7 6 cores and 16 GB RAM machine. The four real-life scientific workflows considered in this experiment are Montage, LIGO, SIPHT and CyberShake. According to the uniform distribution, the sizes of input/output are in the range [10,100] and their compute units are [1,32], respectively. The random values $rd_1$ and $rd_2$ are generated by uniform distribution in the range of [0, 1]. The structure of the scientific workflow is shown in Figure 4. We set the bandwidth to 0.1 G/s if the VMs are located within the same cloud and the bandwidth is 0.05 G/s if the VMs are located within different clouds. Moreover, the failure coefficients of different clouds (Amazon EC2, Microsoft Azure and Google Compute Engine) are $\lambda_1 = 0.001$, $\lambda_2 = 0.003$ and $\lambda_3 = 0.002$, respectively. In the multi-objective problem considered in this paper, the reliability of the workflow should be greater or equal to the reliability constraint in Eq. (22). The maximum reliability is calculated by using Eq. (19).

$$\prod_{i=1}^{n} rel^{max}(t_i) = rel^{max}. \tag{32}$$

FR-MOS algorithm is based on PSO, where $\varphi_1 = \varphi_2 = 2.05$, w $= 0.5$ and the number of particles $N_P = 50$. For the MOS algorithm, the number of compensation solutions $N_S = 25$, the repeat time $N_{IT} = 1000$ and repeat programming is 10 times. To provide sufficient reliability for each workflow, in addition to the maximum workflow reliability, we also calculate the minimum reliability ($rel^{min}$) of workflow. Then, users set workflow reliability constraints as follows:

$$rel_c = rel^{min} + \rho . \left( rel^{max} - rel^{min} \right), \tag{33}$$

where $\rho \epsilon$ [0,1]. According to our proposed algorithm, the value of reliability constraints coefficient $\rho$ can be set

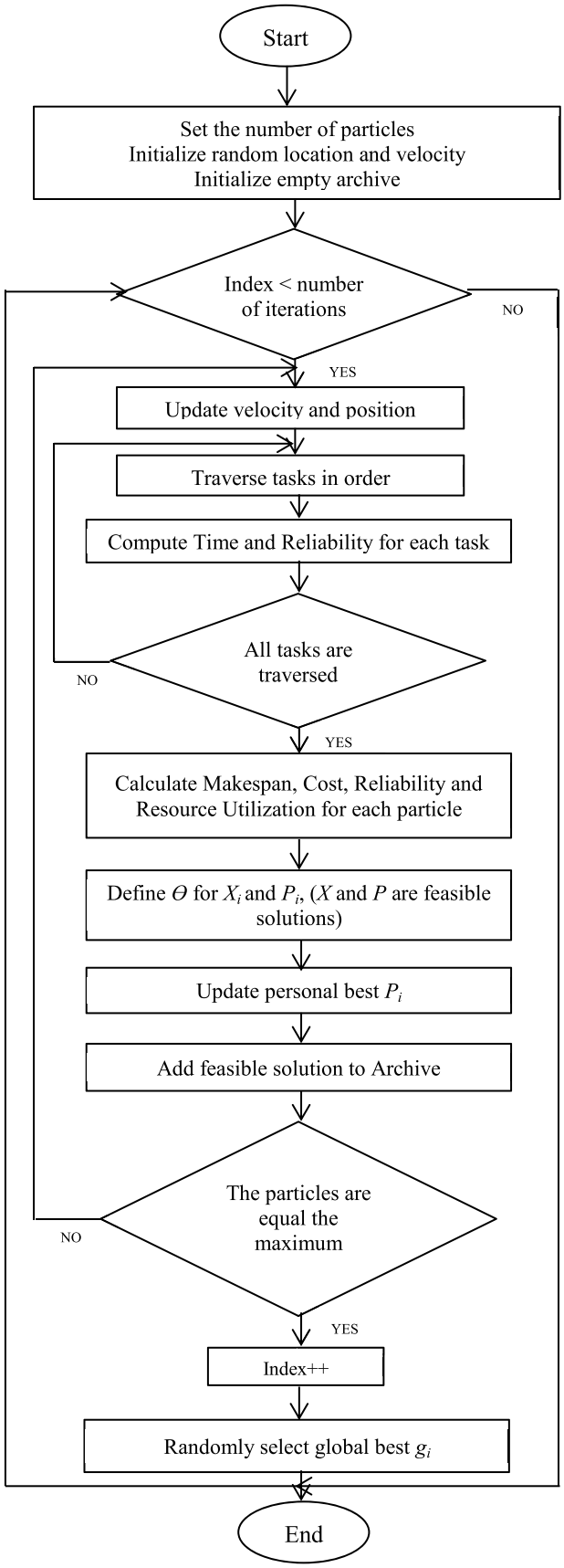



**FIGURE 4.** Flowchart of FR-MOS algorithm.

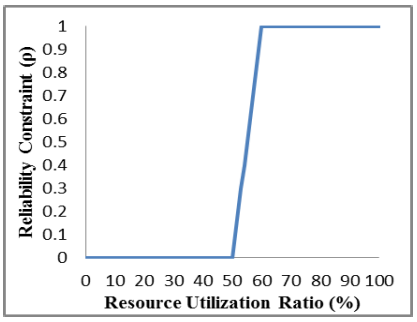



**FIGURE 5.** The relation between resource utilization and reliability constraint coefficient ($\rho$).

based on the result of the fuzzy logic resource utilization [37] by using the following equation.

$$\rho = \begin{cases} 0 & if\ (utilization \leq 50) \\ \dfrac{utilization - 50}{10} & if\ (utilization > 50\ \&\&\ utilization < 60) \\ 1 & otherwise \end{cases} \tag{34}$$

From Eq. (33), the reliability constraint must be in the proper range [$rel^{min}$, $rel^{max}$]. By observing the behaviour of FR-MOS algorithm while scheduling workflow, we observe that the non-constancy between 50 and 60 produce better makespan-cost trade-offs than MOS. Figure 5 shows the relationship between resource utilization and reliability constraint coefficient ($\rho$).

### B. TIME COMPLEXITY ANALYSIS

The WHILE loop in FR-MOS algorithm (lines 7-44) impacts greatly on the time complexity of the algorithm. Each iteration determines the number of positions and velocities, as updated by the dimension of the coding method (lines 9-10). The makespan time complexity and cost depend on the number of tasks, therefore, one particle has $O(n)$ time complexity (lines 11-21). Besides, updating set A with the possible solutions required the worst time $O(Ns)$ (lines 36-39). Task ordering strategy is executed only once for each workflow (see Algorithm 2) and its time complexity could be ignored. Normally, $O(N_{IT}.N_P.n)$ is the time complexity of the FR-MOS algorithm.

### C. SIMULATION RESULTS

Figure 6 shows the makespan-cost trade-offs for MOS and FR-MOS on different workflows.

Generally, FR-MOS algorithm provides the optimal makespan-cost trade-offs compared to the MOS algorithm, in the scientific workflows considered. Table 6 highlights the multi-objective performance metrics in Figure 6. If Q-metric set True, that means the FR-MOS algorithm is better than the MOS algorithm.

Table 6 shows that the value of Q-metrics Q(FR-MOS, MOS) is True for all workflows. This indicates that the Pareto solutions of the FR-MOS are always superior to MOS, which implies the FR-MOS algorithm has the best result in terms of the convergence of multi-objectives. The values of the FS-metric of the FR-MOS algorithm are higher than those of the MOS algorithm for all workflows. For example, the FS-metric value for Montage workflow of FR-MOS is 1.9 (Table 6) in contrast to 1.7 for MOS. This means that FR-MOS produces better diversity than MOS. As regards S-metric, the FR-MOS algorithm has a lower value. For instance, the value of S-metric for SIPHT workflow of the FR-MOS is 0.40 (Table 6), which is smaller than MOS'. This means that FR-MOS algorithm produces better uniformity in the Pareto front than the MOS algorithm.

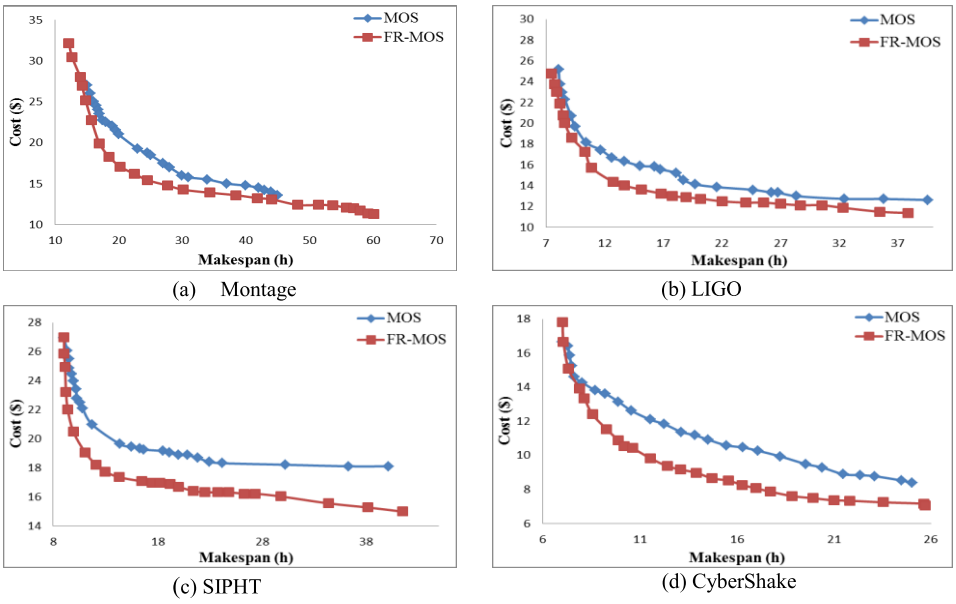
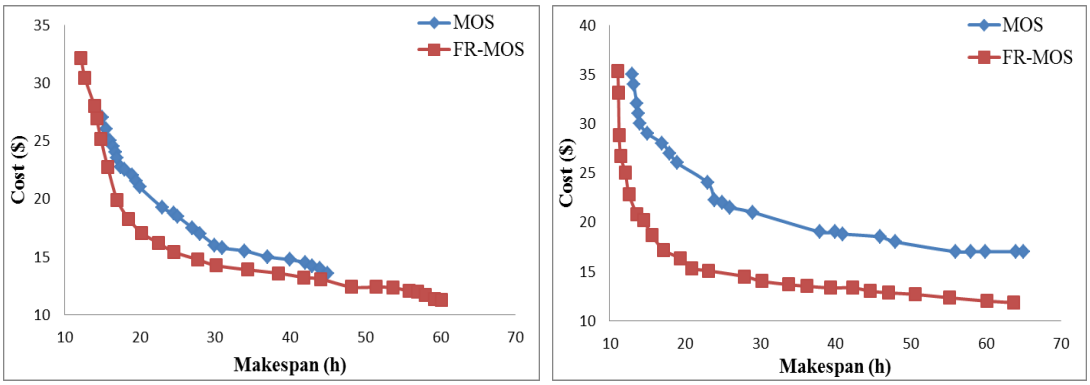
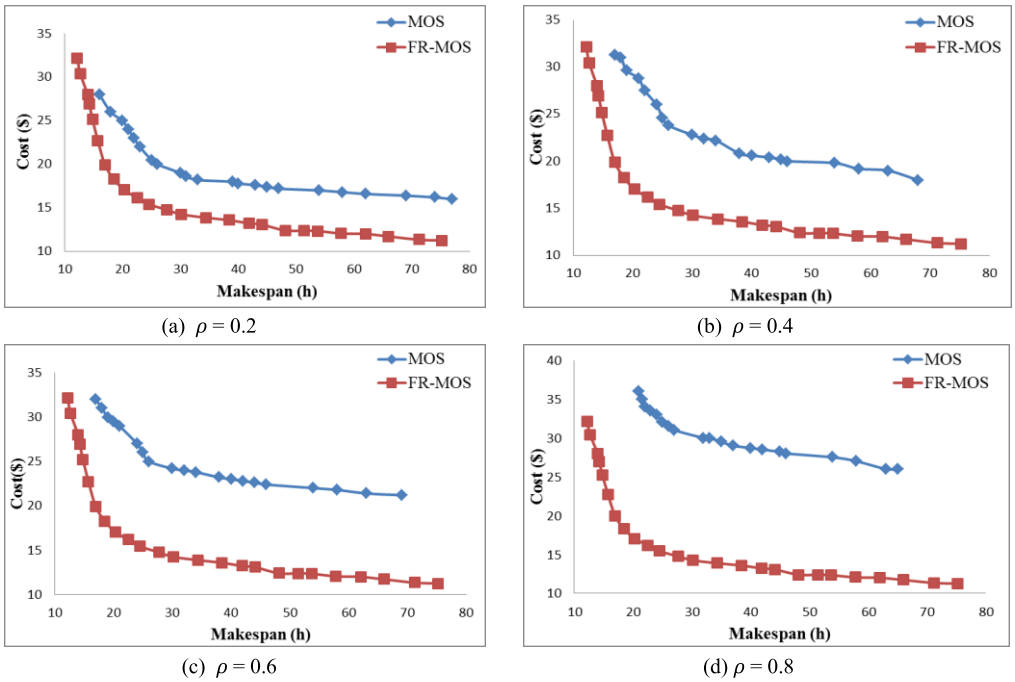



**FIGURE 6.** The trade-offs of makespan-cost on scientific workflow.




(a) Baseline (b) Output data size increase

**FIGURE 7.** Effect of increasing output data size.




(a) $\rho = 0.2$ (b) $\rho = 0.4$

(c) $\rho = 0.6$ (d) $\rho = 0.8$

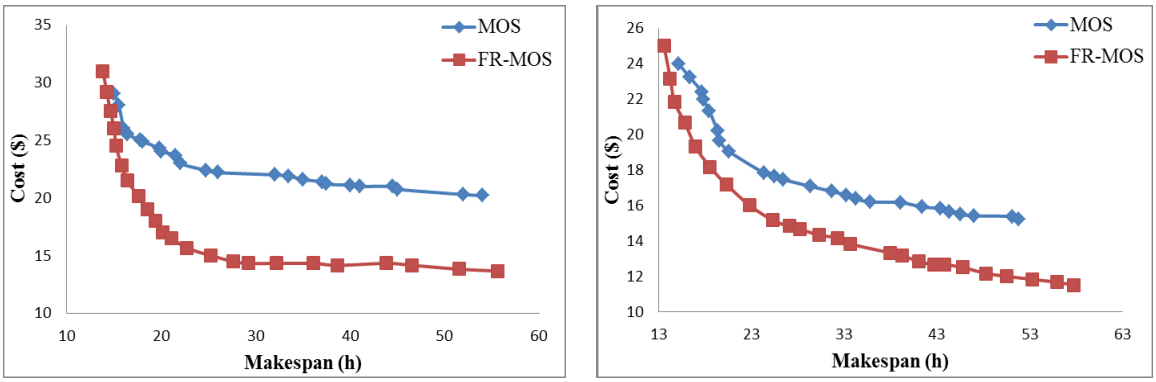
**FIGURE 8.** Trade-offs of makespan-cost for different reliability constraints.

**TABLE 6.** Multi-objective performance metrics.

| Workflow | Q-metric | FR-MOS | MOS |
|---|---|---|---|
| Montage | FR-MOS | - | True |
| | FS-metric | 0.19 | 0.17 |
| | S-metric | 0.24 | 0.47 |
| LIGO | FR-MOS | - | True |
| | FS-metric | 0.40 | 0.34 |
| | S-metric | 0.23 | 0.50 |
| SIPHT | FR-MOS | - | True |
| | FS-metric | 0.32 | 0.17 |
| | S-metric | 0.40 | 0.87 |
| CyberShake | FR-MOS | - | True |
| | FS-metric | 0.42 | 0.39 |
| | S-metric | 0.15 | 0.24 |

In order to sequentially compare the trade-offs in makespan and cost using different output data size and a montage workflow structure-based simulation, the experiment considers the compute unit [1, 32] and output data size [10, 100] as a baseline. On the other hand, the output data size increased to [50, 100]. Figure 7 shows that the simulation results in Figure 7(b) is higher than the results in Figure 7(a). This is because the increasing output data size needs more time to process. Notice that the results of FR-MOS are better than MOS in all cases in Figure 7.

Figure 8 shows the comparison between the FR-MOS algorithm and MOS with different reliability constraints. In the FR-MOS algorithm, we make reliability constraints depend on the fuzzy logic. This is applied to the result of resource utilization so that each workflow has a different reliability constraint associated with its resource utilization according to Equations (16), (17) and (34). From Equations (18) and (19), high workflow reliability was obtained when the rent time of VM is low. However, increasing the reliability constraint makes the workflow's makespan low and increases the cost. Thus, a low VM rent time can be




$[\lambda_1, \lambda_2, \lambda_3] = [0.001, 0.003, 0.002]$ $[\lambda_1, \lambda_2, \lambda_3] = [0, 0, 0]$

**FIGURE 9.** The trade-offs of makespan-cost on different failure coefficient.

obtained when the VM executes tasks with high performance. FR-MOS algorithm can always yield sufficient Pareto solutions and perform better than the MOS algorithm, regardless of its reliability constraints.

We can see in Section 3.F that the workflow reliability depends on the failure coefficient, such as $\lambda_m = 0$ $\{m|m = 1, 2, 3\}$. The reliability of the task will be 1 and it is independent of the rent time of VM. The task $t_i$ chooses the VM with low cost and low performance. The results of two different groups of failure coefficient shown in Figure 9 are $[\lambda_1, \lambda_2, \lambda_3] = [0.001, 0.003, 0.002]$ and $[\lambda_1, \lambda_2, \lambda_3] = [0, 0, 0]$. In the second case, we can see that all the clouds in IaaS platforms are failure-free and the result of its makespan-cost trade-off is better than the first case.

In summary, this study creates three groups of reliability constraints (i.e., high-reliability constraint, low-reliability constraint and non-consistency between them) depending on the resource utilization. With high-reliability constraints, the makespan is always low and the cost is high due to high utilization. For low-reliability constraints, the makespan becomes high and the cost is low. Also, the non-constancy constraint is used to adjust the constraint function when the resource utilization cannot be classified as high or low.

Thus, the disparity between low and high-reliability constraints produced better makespan-cost trade-offs than MOS whose reliability constraint coefficient is defined by users.

## VI. CONCLUSION

In this paper, we proposed a multi-objective scheduling algorithm with fuzzy resource utilization (FR-MOS). The algorithm deploys the particle swarm optimization (PSO) strategy for scheduling real-life scientific workflows in a multi-cloud environment. The main goal of this algorithm is to minimize cost and makespan considering the reliability constraints, where the constraint coefficient is determined by the utilization of cloud resources. This study focused on virtual machines (VMs) of three commercial cloud providers (Amazon EC2, Microsoft Azure and Google compute engine). We noticed that it is essential that users pay more attention to the entire workflow reliability considering the failure coefficient of different cloud platforms while scheduling tasks to VMs. Also, reliability is one of the most significant qualities of service requirements that must be considered by the users when mapping tasks to VMs in different platforms. To solve this problem, our proposed algorithm (FR-MOS) considers the location of executed tasks and their order of data transmission concurrently. Simulation results show that FR-MOS algorithm outperforms the basic MOS algorithm in relation to the multi-objective performance metrics studied. As future work, we will extend our workflow scheduling strategy for achieving fault tolerance in multi-cloud environment. Also we would consider the prospects of scheduling workflows to reduce energy consumption in a hybrid cloud environment.

## REFERENCES

[1] S. Aslam, S. U. Islam, A. Khan, M. Ahmed, A. Akhundzada, and M. K. Khan, "Information collection centric techniques for cloud resource management: Taxonomy, analysis and challenges," *J. Netw. Comput. Appl.*, vol. 100, pp. 80–94, Dec. 2017.

[2] V. Chang and G. Wills, "A model to compare cloud and non-cloud storage of big data," *Future Gener. Comput. Syst.*, vol. 57, pp. 56–76, Apr. 2016.

[3] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "An algorithm for network and data-aware placement of multi-tier applications in cloud data centers," *J. Netw. Comput. Appl.*, vol. 98, pp. 65–83, Nov. 2017.

[4] P. Mell and T. Grance, "Effectively and securely using the cloud computing paradigm," *NIST, Inf. Technol. Lab.*, vol. 2, no. 8, pp. 304–311, Oct. 2009.

[5] G. Sun, D. Liao, D. Zhao, Z. Xu, and H. Yu, "Live migration for multiple correlated virtual machines in cloud-based data centers," *IEEE Trans. Serv. Comput.*, vol. 11, no. 2, pp. 279–291, Mar. 2018.

[6] S. Ghazouani and Y. Slimani, "A survey on cloud service description," *J. Netw. Comput. Appl.*, vol. 91, pp. 61–74, Aug. 2017.

[7] J. Rao, Y. Wei, J. Gong, and C.-Z. Xu, "QoS guarantees and service differentiation for dynamic cloud applications," *IEEE Trans. Netw. Serv. Manage.*, vol. 10, no. 1, pp. 43–55, Mar. 2013.

[8] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," *Future Gener. Comput. Syst.*, vol. 65, pp. 140–152, Dec. 2016.

[9] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, Apr. 2014.

[10] W. Song, F. Chen, H.-A. Jacobsen, X. Xia, C. Ye, and X. Ma, "Scientific workflow mining in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 10, pp. 2979–2992, Oct. 2017.

[11] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," *IEEE Trans. Serv. Comput.*, vol. 11, no. 4, pp. 713–726, Jul. 2018.

[12] B. Lin, W. Guo, G. Chen, N. Xiong, and R. Li, "Cost-driven scheduling for deadline-constrained workflow on multi-clouds," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshop*, May 2015, pp. 1191–1198.

[13] N. Sooezi, S. Abrishami, and M. Lotfian, "Scheduling data-driven workflows in multi-cloud environment," in *Proc. IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. (CloudCom)*, Nov. 2015, pp. 163–167.

[14] I. Gupta, M. S. Kumar, and P. K. Jana, "Compute-intensive workflow scheduling in multi-cloud environment," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Sep. 2016, pp. 315–321.

[15] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities," *J. Netw. Comput. Appl.*, vol. 68, pp. 173–200, Jun. 2016.

[16] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1344–1357, May 2016.

[17] E. Jeannot, E. Saule, and D. Trystram, "Optimizing performance and reliability on heterogeneous parallel systems: Approximation algorithms and heuristics," *J. Parallel Distrib. Comput.*, vol. 72, no. 2, pp. 268–280, Feb. 2012.

[18] S. Hwang and C. Kesselman, "Grid workflow: A flexible failure handling framework for the grid," in *Proc. 12th IEEE Int. Symp. High Perform. Distrib. Comput.*, Jan. 2004, pp. 126–137.

[19] D. Poola, K. Ramamohanarao, and R. Buyya, "Enhancing reliability of workflow execution using task replication and spot instances," *ACM Trans. Auton. Adapt. Syst.*, vol. 10, no. 4, pp. 1–21, Feb. 2016.

[20] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *J. Netw. Comput. Appl.*, vol. 79, pp. 88–115, Aug. 2017.

[21] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *J. Netw. Comput. Appl.*, vol. 114, pp. 108–122, Jul. 2018.

[22] S. Singh and I. Chana, "QoS-aware autonomic resource management in cloud computing: A systematic review SUKHPAL," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–46, 2016.

[23] R. N. Calheiros and R. Buyya, "Meeting deadlines of scientific workflows in public clouds with tasks replication," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1787–1796, Jul. 2014.

[24] L. Zeng, B. Veeravalli, and A. Y. Zomaya, "An integrated task computation and data management scheduling strategy for workflow applications in cloud environments," *J. Netw. Comput. Appl.*, vol. 50, pp. 39–48, Apr. 2015.

[25] L. Zeng, B. Veeravalli, and X. Li, "SABA: A security-aware and budget-aware workflow scheduling strategy in clouds," *J. Parallel Distrib. Comput.*, vol. 75, pp. 141–151, Jan. 2015.

[26] J. J. Durillo, H. M. Fard, and R. Prodan, "MOHEFT: A multi-objective list-based method for workflow scheduling," in *Proc. 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, Dec. 2012, pp. 185–192.

[27] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Inf. Sci.*, vol. 379, pp. 241–256, Feb. 2017.

[28] S. Kianpisheh, N. M. Charkari, and M. Kargahi, "Ant colony based constrained workflow scheduling for heterogeneous computing systems," *Cluster Comput.*, vol. 19, no. 3, pp. 1053–1070, Sep. 2016.

[29] N. Kaur and S. Singh, "A budget-constrained time and reliability optimization bat algorithm for scheduling workflow applications in clouds," *Procedia Comput. Sci.*, vol. 98, pp. 199–204, 2016.

[30] D. Hutchison and J. C. Mitchell, "Swarm, evolutionary, and memetic computing," in *Proc. 2nd Int. Conf. SEMCCO*, Visakhapatnam, India, Dec. 2011, pp. 16–18.

[31] R. Garg and A. K. Singh, "Multi-objective workflow grid scheduling using $\varepsilon$-fuzzy dominance sort based discrete particle swarm optimization," *J. Supercomput.*, vol. 68, no. 2, pp. 709–732, May 2014.

[32] I. Casas, J. Taheri, R. Ranjan, and A. Y. Zomaya, "PSO-DS: A scheduling engine for scientific workflow managers," *J. Supercomput.*, vol. 73, no. 9, pp. 3924–3947, Sep. 2017.

[33] 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.

[34] C. Zhang, R. Green, and M. Alam, "Reliability and utilization evaluation of a cloud computing system allowing partial failures," in *Proc. IEEE 7th Int. Conf. Cloud Comput.*, Jun. 2014, pp. 936–937.

[35] S. Kianpisheh, N. M. Charkari, and M. Kargahi, "Reliability-driven scheduling of time/cost-constrained grid workflows," *Future Gener. Comput. Syst.*, vol. 55, pp. 1–16, Feb. 2016.

[36] H. M. Fard, R. Prodan, and T. Fahringer, "Multi-objective list scheduling of workflow applications in distributed computing infrastructures," *J. Parallel Distrib. Comput.*, vol. 74, no. 3, pp. 2152–2165, Mar. 2014.

[37] C. V. Negoita, "Fuzzy sets," *Fuzzy Sets Syst.*, vol. 133, no. 2, p. 275, Jan. 2003.

[38] J. Mendel, "Fuzzy logic systems for engineering: A tutorial," *Proc. IEEE*, vol. 83, no. 3, pp. 345–377, Mar. 1995.

[39] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend, "A MOPSO algorithm based exclusively on Pareto dominance concepts," in *Proc. Int. Conf. Evol. Multi-Criterion Optim.*, 2005, pp. 459–473.

[40] M. Cafaro, G. Aloisio, G. Juve, and E. Deelman, *Grids, Clouds and Virtualization*. London, U.K.: Springer, 2011, pp. 71–91.

[41] K. Deb, "An efficient constraint handling method for genetic algorithms," *Comput. Methods Appl. Mech. Eng.*, vol. 186, nos. 2–4, pp. 311–338, Jun. 2000.

[42] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.

[43] J. Wei and M. Zhang, "A memetic particle swarm optimization for constrained multi-objective optimization problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2011, pp. 1636–1643.

[44] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 348–365, Jun. 2014.

[45] Z. Zhang, L. Cherkasova, and B. T. Loo, "Optimizing cost and performance trade-offs for MapReduce job processing in the cloud," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–8.

[46] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Hum. Sci.*, vol. 7803, 1995, pp. 39–43.

[47] W.-F. Leong and G. Yen, "PSO-based multiobjective optimization with dynamic population size and adaptive local archives," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 38, no. 5, pp. 1270–1293, Oct. 2008.

[48] H.-P. Dai, D.-D. Chen, and Z.-S. Zheng, "Effects of random values for particle swarm optimization algorithm," *Algorithms*, vol. 11, no. 2, p. 23, Feb. 2018.

[49] Y. D. Valle, G. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 171–195, Apr. 2008.

[50] W. Jing, Z. Yongsheng, Y. Haoxiong, and Z. Hao, "A trade-off Pareto solution algorithm for multi-objective optimization," in *Proc. 5th Int. Joint Conf. Comput. Sci. Optim.*, Jun. 2012, pp. 123–126.

[51] L. Wang and K. Chen, "Advances in natural computation," in *First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings*. Springer, 2005.

**MAZEN FARID EBRAHIM** received the bachelor's degree in computer science and engineering from the University of Aden, and the master's degree in computer science from Universiti Putra Malaysia, where he is currently pursuing the Ph.D. degree with the Faculty of Computer Science and Information Technology. His research interests fall under computer networks, workflow scheduling, and cloud computing.

**ROHAYA LATIP** received the bachelor's degree in computer science from the University Technology Malaysia, Malaysia, in 1999, and the M.Sc. degree in distributed systems and Ph.D. degree in distributed database from University Putra Malaysia. She is currently the Head of the Department of Communication Technology and Network, where she is also an Associate Professor with the Faculty of Computer Science and Information Technology. She is also the Head of the HPC Section, University Putra Malaysia, from 2011 to 2012, and consulted the Campus Grid project and also the Wireless for hostel in Campus UPM project. She is also a Co-Researcher at the Institute for Mathematic Research (INSPEM). Her research interests include big data, cloud and grid computing, network management, and distributed database.

**MASNIDA HUSSIN** (Member, IEEE) received the Ph.D. degree from The University of Sydney, Australia, in 2012. She is currently an Associate Professor with the Department of Communication Technology and Network, Faculty of Computer science and Information Technology, Universiti Putra Malaysia, Malaysia. Her main research interests are in QoS and resource management for distributed systems such as grid and cloud. She was also involved in green computing project. She received Huawei Technology Certification, in 2012, as a System Instructor, that makes her specialized in configuring Huawei network computer components. She has also published several papers that are related to parallel and distributed computing.

**NOR ASILAH WATI ABDUL HAMID** received the Ph.D. degree from the University of Adelaide, in 2008. In 2011, she did her postdoctoral research at the High Performance Computing Lab, The George Washington University, Washington, D.C., USA. She is currently an Associate Professor with the Faculty of Computer Science and Information Technology, University Putra Malaysia, where she is also an Associate Researcher of high speed machine with the Institute for Mathematical Research (INSPEM). Her research interests are in parallel and distributed high performance computing, cluster computing, computational science, and other applications of high-performance computing.