

RESEARCH ARTICLE

Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments

DINESHAN SUBRAMONEY AND CLEMENT N. NYIRENDA 

Department of Computer Science, University of the Western Cape, Cape Town 7535, South Africa

Corresponding author: Clement N. Nyirenda (cnyirenda@uwc.ac.za)

This work was supported in part by Telkom South Africa through the Centre of Excellence Program at the University of the Western Cape.

ABSTRACT Scientific workflow scheduling involves the allocation of workflow tasks to particular computational resources. The generation of optimal solutions to reduce run-time, cost, and energy consumption, as well as ensuring proper load balancing, remains a major challenge. Therefore, this work presents a Multi-Swarm Particle Swarm Optimization (MS-PSO) algorithm to improve the scheduling of scientific workflows in cloud-fog environments. MS-PSO seeks to address the canonical PSO's problem of premature convergence, which leads it to suboptimal solutions. In MS-PSO, particles are divided into several swarms, with each swarm having its own cognitive and social learning coefficients. This work also develops a weighted sum objective function for the workflow scheduling problem, based on four objectives: makespan, cost, energy and load balancing for cloud and fog tiers. The FogWorkflowSim Toolkit is used in the evaluation process, with the objectives serving as performance metrics. The MS-PSO approach is compared with the canonical PSO, Genetic Algorithm (GA), Differential Evolution (DE) and GA-PSO. The following scientific workflows are used in the simulations: Montage, Cybershake, Epigenomics, LIGO and SIPHT. MS-PSO outperforms the canonical PSO on all scientific workflows and under all performance metrics. It competes fairly well against the other approaches and it is more stable and reliable. It only ranks second to PSO, in terms of execution time. In future, multiple species, incorporating population update mechanisms from several algorithmic frameworks (MS-PSO, DE, GA), will be used for scientific workflow scheduling. Hybridization of the realized algorithm with dynamic approaches will also be investigated.


INDEX TERMS Scientific workflows, cloud computing, fog computing, particle swarm optimization, evolutionary algorithms.

I. INTRODUCTION

A *scientific workflow* is the description of a process for accomplishing a scientific objective, which is expressed in terms of tasks and their dependencies [1]. These dependencies occur at various stages and the tasks are executed in a predefined order, in order to achieve the scientific objective [2]. Scientific workflows are typically described as a directed acyclic graph (DAG), where the nodes are denoted as tasks and the edges, as task dependencies [3]. The concept of scientific workflows has largely been triggered by the emergence of data-intensive and computationally complex

methods in the natural sciences and the need for techniques for the automation of recurring computational tasks [4].

Early efforts on scientific workflows were implemented on High Performance Computing (HPC) [5] and distributed systems [6], [7]. In most of these early implementations, the systems and the applications were considered as black boxes; the main focus was on distributed resource management, workload, and execution management [6]. In recent decades, scientific workflows are being implemented on cloud computing infrastructure [8], [9], [10]. Unlike distributed systems, cloud computing is client-server based architecture, where resources are utilized in a centralized manner on a pay-as-you-go model [11]. Cloud computing has become the platform of choice for the execution of computation-intensive scientific workflows [3], [9], [10], because it avails the

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

following positive attributes: cost efficiency, high speed, excellent accessibility, manageability, elasticity, virtualisation capabilities and sporadic batch processing.

Scheduling workflows for execution on cloud resources involves the mapping of the tasks in the workflows to the available virtual machines in the cloud infrastructure. This process is, generally, hindered by high computation and communication costs [12]. As a result, population-based algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) have been proposed for scheduling tasks or workflows on the cloud infrastructure [12], [13], [14], [15]. These approaches can be generally described as static workflow scheduling mechanisms because they do the mapping process in an *a priori* manner. The major optimization objectives in these approaches are maximization of the execution finishing time, commonly known as makespan, and minimization of the cost. Other quality of service (QoS) requirements such as deadline and budget are usually incorporated as constraints.

Two recent surveys on PSO-based scheduling techniques for scientific workflows clearly show that PSO is the most popular population-based optimization approach in this field [15], [16]. The PSO technique is preferred because of its fast convergence and short run time. Most of the PSO-based works use the canonical PSO [17], which suffers from premature convergence [18], [19], [20]. In recent years, new PSO variants known as multi-swarm PSOs, that divide the population into sub-swarms in the search space, have been proposed and evaluated on standard test functions. These new techniques have exhibited better performance than the canonical approach due to their superior exploration capabilities [19], [20], [21], [22], [23], [24], [25], [26], [27]. It, therefore, naturally follows that there is a high likelihood for a multi-swarm PSO to exhibit better performance even in the scientific workflow scheduling problem. This forms the motivation behind the work presented in this paper.

Another important observation is that most of the scientific scheduling approaches in literature are implemented on the traditional two-tier architecture, with the cloud servers at the top and the end devices at the low level. WorkflowSim [28] has been the dominant simulation platform. Nevertheless, workflow execution on the cloud is hampered by increased latency and low QoS [29], [30]. In order to address these limitations, a three-tier network that incorporates the fog layer between cloud servers and end devices has been proposed as the demand for cloud services grows. In this new network, fog devices extend the cloud computing model by executing some tasks thereby reducing the execution overhead on the cloud infrastructure. This helps to reduce latency, improve user experience, enhance data security, and improve energy efficiency. Currently, very few works in scientific workflow scheduling [18], [31], [32] have focused on this emerging paradigm.

This paper adopts the three-tier network. The preliminary version of this work was presented and published at a conference [32]. This work is implemented on the recently

proposed FogWorkflowSim [33]. The major contributions of the work proposed in this paper are as follows:

- 1) The addition of load balancing to the weighted sum objective function that already incorporates makespan, cost, and energy consumption, as presented in [32].
- 2) A review of the state-of-the-art multi-swarm PSO algorithms and techniques.
- 3) The proposal for Multi-Swarm PSO algorithm for scientific workflow scheduling and its implementation in FogWorkflowSim [33].
- 4) A comparative evaluation of MS-PSO, GA, PSO, GA-PSO and DE as scheduling algorithms for scientific workflows is conducted with makespan, cost, load balancing, and energy as performance metrics. The number of tasks has been increased to a maximum of 500 in each case.

The rest of this paper is organized as follows. Section II presents the state-of-the-art on multi-swarm PSO algorithms. Section III presents concept of scientific workflows. Section IV outlines the workflow optimization process while section V presents the proposed MS-PSO algorithm. Section VI presents the simulation environment and the performance evaluation. Finally, section VII concludes the paper.

II. STATE OF THE ART ON MULTI-SWARM PSO APPROACHES

Before making a brief presentation on some of the most prominent multi-swarm PSO algorithms, the canonical PSO is presented briefly.

A. THE CANONICAL PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle Swarm Optimization (PSO) is a population-based stochastic optimization algorithm introduced by Kennedy and Eberhart [17]. The technique is used to solve optimization problems by emulating the social behavior of bird flocking, fish schooling and other animal societies that cooperate and share information to improve their position without relying on a leader. The PSO algorithm has received more and more attention from many researchers due to its relative simplicity and fast convergence [18]. Over the years, the PSO algorithm has been successfully applied in a variety of fields, such as constrained mixed-variable optimization problems [34] and wireless sensor networks [35].

This technique uses a population of N individuals, represented as particles in the H -dimensional solution space. The current position of the i -th particle is represented as $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,H})$ and its velocity is represented as $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \dots, v_{i,H})$. The quality of each particle is measured using a defined fitness function depending on the optimization problem. Each particle's movement is based on its best known personal position \mathbf{pBest}_i , and also moves towards the best known global position \mathbf{gBest} for the entire swarm. This process leads the swarm to the best position over a number of iterations in the search process. At the k -th

iteration, the elements of the particle's velocity and position are updated by using

$$v_{i,h} = \omega v_{i,h} + c_1 r_1 (\text{pBest}_{i,h} - x_{i,h}) + c_2 r_2 (\text{gBest}_h - x_{i,h}) \quad (1)$$

$$x_{i,h} = x_{i,h} + v_{i,h}, \quad (2)$$

where $h = 1, 2, \dots, H$; ω is the inertia weight; r_1 and r_2 are random numbers between (0,1); and c_1 and c_2 are the learning factors.

B. MULTI-SWARM PSO APPROACHES

In recent years, there have been many efforts to break down the PSO algorithm into sub-swarms, and a brief overview of some of the most notable ones is presented as follows.

a) **Dynamic multi-swarm PSO [21]**: This is arguably the first highly notable multi-swarm PSO approach. It divides the population into many small dynamic swarms which are evolved in a similar fashion; these swarms are regrouped frequently thereby exchanging information among the swarms. This multi-swarm PSO exhibited better performance than several PSO variants on a set of shifted rotated benchmark functions.

b) **The Socio-Cognitively Inspired PSO [22]**: This approach divides the PSO swarm into sub-swarms, called species because they have unique evolution mechanisms. The particles get inspired by the global and the local optima, but they also share their knowledge of optimal locations with neighboring particles belonging to other species. Experiments were conducted with various proportions of different species in the population to find the best setting.

c) **HCLDMS-PSO [23]**: The heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators (HCLDMS-PSO) modifies the traditional dynamic multi-swarm PSO [21] by adding a comprehensive learning (CL) strategy, where the global optimal experience of the whole population is conducted to generate an exploitation sub-population exemplar.

d) **HIDMS-PSO [24]**: The Heterogeneous Improved Dynamic Multi-Swarm PSO (HIDMS-PSO) is also an extension of the traditional dynamic multi-swarm PSO [21]. The population is initially divided into two sub-populations, with the first one being further divided into sub-swarms. The particles in these sub-swarms are guided heterogeneously. On the other hand, the second sub-population is guided homogeneously by using the classical PSO update mechanism.

e) **DMS-GPSO [25]**: The dynamic multi-swarm global particle swarm optimization (DMS-GPSO) segments the evolutionary process into an initial stage and a final stage. In the initial stage, the population is divided into a global sub-swarm and multiple dynamic sub-swarms. The global sub-swarm focuses on exploitation, guided by the globally optimal particle. On the other hand, multiple sub-swarms focus on exploration, guided by the best particle in the neighborhood. In the final stage, the elite particles stored in

an archive are combined with the DMS sub-swarms to form a single population with the aim of improving exploitation capabilities.

f) **DMS-PSO-GD [19]**: The Dynamic Multi-Swarm Particle Swarm Optimization with Global Detection Mechanism (DMS-PSO-GD) has some similarities to the DMS-GPSO [25] in the sense that it also divides the population into two kinds of sub-swarms: several same-sized dynamic sub-swarms and a global sub-swarm. This algorithm, however, uses the variances and average fitness values of dynamic sub-swarms to measure the distribution of the particles, in order to detect the dominant and the optimal particle.

g) **PSOMAS [26]**: In the Particle Swarm Optimization with Multiple Adaptive Sub-swarms (PSOMAS) algorithm, each sub-swarm is evolved by a completely different variant of the single swarm PSO algorithm, such as Comprehensive Learning PSO [36] and Cooperative PSO [37]. This work also uses an adaptive strategy to reduce usage of computational resources.

h) **Chain and Hypercube Communication Topologies for multiswarm-PSOs [27]**: The chain and hypercube topologies have been proposed with the aim of limiting communication between swarms in order to increase per-swarm exploration over different parts of the search space. A comparison of these techniques with the simple cross-over strategy showed that the chain topology exhibited a better error performance. The computational complexities of these techniques is, however, not discussed.

The multi-swarm PSO proposed in this work is generally inspired by the Socio-Cognitively Inspired PSO [22]. This algorithm will be presented in detail in Section V. The next couple of sections will focus on the application domain - the scientific workflow scheduling, with the aim of developing the objective function that will be used in the multi-swarm PSO scheduling algorithm.

III. THE CONCEPT OF WORKFLOWS

The workflow application is modelled as a Direct Acyclic Graph (DAG), defined by $G = (T, E)$, where T refers to the vertices and represents the set of n tasks $\{t_1, t_2, \dots, t_n\}$ and E is the set of directed edges, which indicates the dependencies or precedence constraints between tasks in the workflow [13], [14], [18], [32]. An edge can also be illustrated in terms of inter-task data, $d_{i,j} = \langle t_i, t_j \rangle \in E$, where $d_{i,j}$ is a positive value representing the length of the output data from task t_i to be used as input for task t_j . Therefore, the execution of t_j can only begin after the execution of task t_i has completed. A task t_i with no parent is known as a start task and a task t_j with no child is known as an end task. Fig. 1 shows a sample structure of a workflow model. The tasks of the workflow placed horizontally on the same level can execute in parallel order on the cloud-fog system. In this example, tasks t_2, t_3 and t_4 can execute in a parallel order.

Workflow application scheduling in a cloud-fog environment is defined here as the problem of assigning computing resources with different characteristics to tasks

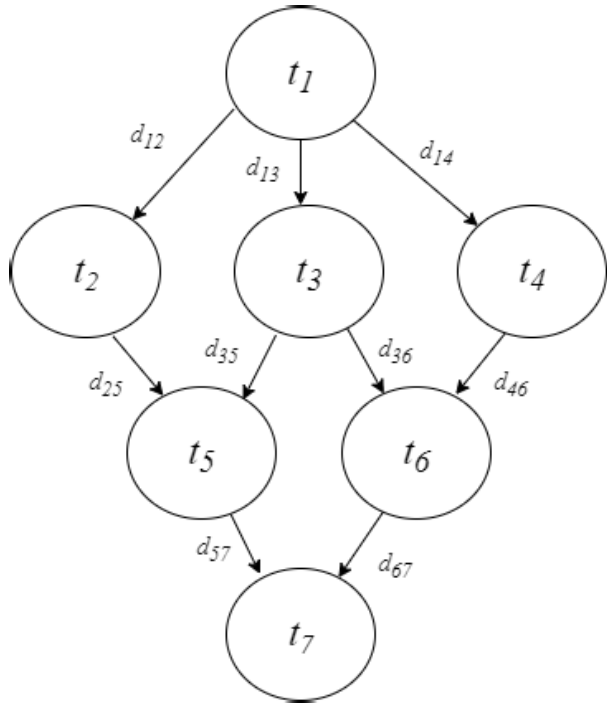


FIGURE 1. Sample workflow model with seven tasks.

of the workflow application, in order to minimize the total completion time, cost and energy of the workflow execution while ensuring that the overall load assigned to each VM is also balanced.

IV. THE PROPOSED WORKFLOW SCHEDULING MULTI-OBJECTIVE FUNCTION FOR THE CLOUD-FOG MODEL

In the cloud-fog environment setup presented in this work, computational resources are of three types, namely cloud servers, fog servers and end devices, as illustrated in Fig. 2. Each of these resources consists of computing and storage capabilities along with memory, bandwidth and power requirements. Computational resources in the cloud and fog are represented as virtual machines (VMs). The end device is included because for some small tasks, transferring them to the fog and cloud servers does not make sense from an economic and resource utilization perspective.

In this work, the selection of the optimal resource is determined by four objectives for optimization process: makespan, cost, energy consumption and load of a VM on the respective cloud and fog layers.

A. MAKESPAN

The makespan of a workflow can be defined as the total execution time for the completion of an entire workflow. Thus, makespan MS is calculated as follows:

$$MS = \max\{FT_{t_i}, t_i \in T\} - \min\{ST_{t_i}, t_i \in T\} \quad (3)$$

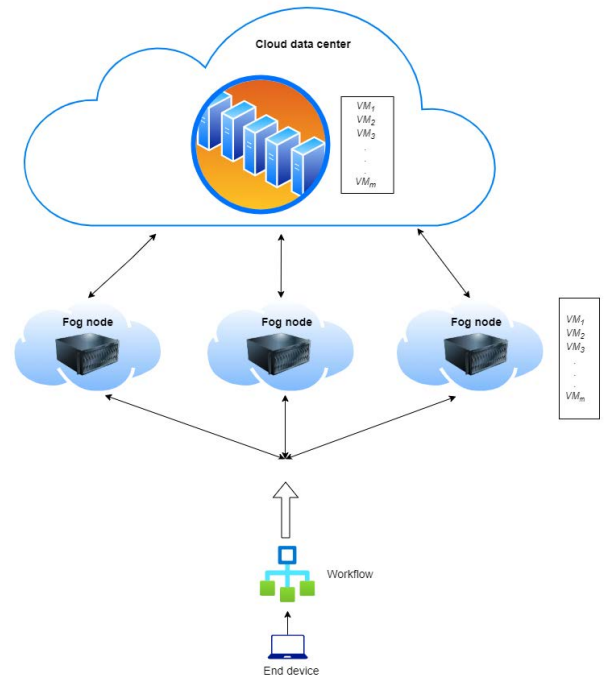


FIGURE 2. Cloud-fog paradigm for workflow scheduling.

where ST_{t_i} and FT_{t_i} are the starting time and finishing time respectively for task t_i in a particular workflow.

B. COST

This consists of computation and communication costs. Computation costs apply for all the three computational resource types while communication costs are not included when the tasks are executed on the end device. The computational cost [18] when using computing resource r is defined as

$$CE_i^r = pr \cdot (FT_{t_i} - ST_{t_i}), \quad (4)$$

where pr is the unit processing cost. The communication cost, which denotes the data transfer cost of a task output of size $d_{i,j}$ from the resource processing task i to the resource allocated to process task j , is determined by

$$CC_{i,j} = trc_{i,j} \cdot d_{i,j}, \quad (5)$$

where $trc_{i,j}$ is the unit cost of communication from the resource, where task i is mapped, to the resource, where task j is mapped; $trc_{i,j} = 0$ when the two tasks are executed on the same resource. Therefore, for m computational resources and n tasks, the total cost TC is

$$TC = \sum_{i=1}^n \sum_{j=1}^n CC_{i,j} + \sum_{r=1}^m \sum_{i=1}^n CE_i^r. \quad (6)$$

C. ENERGY CONSUMPTION

The energy consumption model [38] is constructed using active and idle components, which are denoted by E_{active} and

E_{idle} respectively. The former refers to the energy consumed when a task is being executed while the latter is the energy used when a resource is idling. The active energy is calculated by

$$E_{active} = \sum_{i=1}^n \alpha f_i v_i^2 (FT_{t_i} - ST_{t_i}), \quad (7)$$

where α is a constant; f_i and v_i are the frequency and supply voltage for the resource on which task i is being executed. During the idle period, the resource goes into sleep mode, where the voltage supply level and the relative frequency are at the lowest level. Therefore, the energy consumed during the idle period is determined by using [32] and [38]:

$$E_{idle} = \sum_{j=1}^m \sum_{idle_{j,k} \in IDLE_{j,k}} \alpha f_{min j} v_{min j}^2 L_{j,k}, \quad (8)$$

where $IDLE_{j,k}$ is a set of idling slots on resource j , $f_{min j}$ and $v_{min j}$ refer to the frequency and lowest supply voltage on resource j respectively; $L_{j,k}$ is the duration of idling time for $idle_{j,k}$. The total energy TE consumed on the cloud-fog system for the execution of the entire workflow is

$$TE = E_{active} + E_{idle}. \quad (9)$$

D. LOAD BALANCING

Load balancing of the respective cloud and fog layers is considered in this work. Load balancing can be defined as the calculation of the standard deviation of the load of all the VMs. The aim is to ensure that all the VMs have almost equal loads. So, minimizing the standard deviation of the VMs produces improved load management among the VMs. The load of a VM is determined by the ratio between the length or size of the tasks executed by a VM and capacity of the VM. The fog layer VM characteristics are different compared to the cloud. Therefore to ensure fair overall load distribution, the standard deviation of the load is calculated independently for the cloud and fog layers. The end device is excluded when balancing the load on the cloud-fog environment.

Let c and f denote the number of VMs in the cloud and fog layers respectively. Hence, load balancing on the cloud is defined by

$$LBC = \sqrt{\frac{\sum_{i=1}^c (LC_i - ALC)^2}{c}} \quad (10)$$

where LC_i refers to the load of the i -th VM on the cloud, and ALC is the average load of all the VMs on the cloud layer. Similarly, the load balancing on the fog is defined by

$$LBF = \sqrt{\frac{\sum_{i=1}^f (LF_i - ALF)^2}{f}} \quad (11)$$

where LF_i refers to the load of the i -th VM on the fog, and ALF is the average load of all the VMs on the fog

layer. Therefore, using the four aforementioned objectives, the weighted sum objective function is defined by:

$$F(\mathbf{p}) = w_1 \cdot MS_{norm} + w_2 \cdot TC_{norm} + w_3 \cdot TE_{norm} + w_4 \cdot LBC_{norm} + w_5 \cdot LBF_{norm} \quad (12)$$

where \mathbf{p} is the assignment of the n tasks of a workflow to the m available computing resources. In PSO parlance, \mathbf{p} is referred to as a particle, while in the Genetic Algorithm (GA), it is referred to as the chromosome and in Differential Evolution (DE), it is a vector. Parameters MS_{norm} , TC_{norm} , TE_{norm} , LBC_{norm} , and LBF_{norm} are the normalized makespan, total cost, energy consumption, load balancing among cloud VMs, and load balancing among fog nodes respectively; w_* is the coefficient weight. Equal weights are used in the performance evaluations to obtain an equal contribution of each objective i.e. $w_* = 0.2$. Normalization removes any biases in the objective function and also ensures that there is a balanced contribution from all the objectives [32]. To obtain the normalized objective function parameters in equation (12), objective function values MS_i , TC_i , TE_i , LBC_i and LBF_i are defined for the i^{th} potential solution. The normalized objectives are defined by

$$\begin{aligned} MS_{norm} &= \frac{MS_i - MS_{min}}{MS_{max} - MS_{min}} \\ TC_{norm} &= \frac{TC_i - TC_{min}}{TC_{max} - TC_{min}} \\ TE_{norm} &= \frac{TE_i - TE_{min}}{TE_{max} - TE_{min}} \\ LBC_{norm} &= \frac{LBC_i - LBC_{min}}{LBC_{max} - LBC_{min}} \\ LBF_{norm} &= \frac{LBF_i - LBF_{min}}{LBF_{max} - LBF_{min}} \end{aligned} \quad (13)$$

where the max and min values represent the maximum and minimum of the objectives; these are determined from the initial population of the algorithms.

Now that this section has defined the objective function (Eq. (12)), the next section presents the multi-objective workflow scheduling based on the proposed multi-swarm PSO optimization algorithm.

V. MULTI-OBJECTIVE WORKFLOW SCHEDULING BASED ON MULTI-SWARM PSO OPTIMIZATION ALGORITHM

Before delving into the philosophy of the proposed multi-swarm PSO, the particle \mathbf{p} , which has been introduced in multi-objective function (Eq. (12)) is formally presented in the first subsection. Thereafter, the proposed multi-swarm PSO is presented and illustrated in the second subsection.

A. REPRESENTATION OF THE PARTICLE

As mentioned in previous sections, the workflows in this work can be scheduled for execution at the source end device, at a fog VM or at a cloud VM. The end devices do not offload their tasks to fellow end devices; they can only

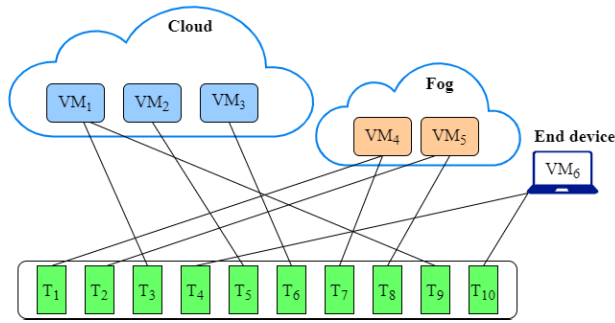


FIGURE 3. Example of a task-resource mapping on the cloud-fog environment.

offload their tasks to fog and cloud resources. Therefore, in the scheduling of workflows, only one representative end device is incorporated in the encoding process. As the scheduling of workflow tasks in a cloud-fog environment is a discrete problem, natural numbers are used to encode the individuals for the proposed Multi-Swarm PSO algorithm. The individuals, referred to as particles, are mappings of task-resource schedules. The dimension or length of each particle \mathbf{p} is n ; this is the total number of tasks in the workflow. Each position of the particle is a positive integer representing the task number. The value assigned to this position is a VM ID that is assigned to execute the task. The ID numbers are selected from the VMs available on the respective architecture tier. Fig. 3 shows a graphical illustration of an example workflow with 10 tasks to be mapped to a cloud-fog setup with 3 cloud VMs and 2 fog VMs. In this example the particle is denoted as $\mathbf{p} = (4, 5, 1, 6, 2, 3, 4, 5, 1, 6)$.

B. THE PROPOSED MS-PSO FOR SCIENTIFIC WORKFLOW SCHEDULING

Scientific workflow scheduling is a computationally complex exercise. Therefore, the multi-swarm PSO optimization algorithm tailored for this problem must be simple and characterized by a low computational complexity. In light of this understanding, this work adopts the Socio-Cognitively Inspired PSO [22] due to its apparent simplicity. Furthermore, in this approach, the swarms or species are evolved differently from each other. The particles’ position and velocity are updated by the different rules specific to a swarm. They, however, exchange information with the neighboring swarms. This behavior is very intuitive from natural point of view because in the real world, a variety of different species of a particular animal usually exist in an ecosystem simultaneously.

The competition for territory and survival creates the need for inter-swarm communication. Therefore, the evolution of the individuals is affected by other swarms’ rules thereby promoting information sharing among swarms and cooperative development of the all species. In the proposed MS-PSO algorithm, each swarm is initialized with particles that belong to a particular type of species. The species in

TABLE 1. Settings of the cognitive and social learning coefficients for different species.

Species Type	Particle (A)	Swarm (B)	All Swarms (C)
Normal	1	1	0
Local & Global	1	0	1
Swarm only	0	1	0
Global only	0	0	1
Random	random	random	random

each swarm has its own algorithm for calculating the velocity. In stead of using the neighborhood’s best position in the third term as in [22], entire swarm’s best position is used. Therefore, at every k -th iteration, the elements of velocity v_i^s of the i -th particle in the s -th swarm are updated by using

$$v_{i,h}^s = \omega v_{i,h}^s + A(\mathbf{pBest}_{i,h}^s - x_{i,h}) + B(\mathbf{sBest}_h^s - x_{i,h}) + C(\mathbf{gBest}_h - x_{i,h}), \quad (14)$$

where \mathbf{pBest}_i^s is known as the personal best position for the i -th particle in the s -th swarm; \mathbf{sBest}^s is the best position in the s -th swarm and \mathbf{gBest} is the global best position found across all swarms. A , B and C are the learning factor coefficients in the range $[0,1]$ which are different for each swarm according to its class of species. The elements of position \mathbf{x}_i^s of the i -th particle are updated by using

$$x_{i,h}^s = x_{i,h}^s + v_{i,h}^s, \quad (15)$$

The five types of species, considered in this work, are presented as follows:

- *Normal Species*: This species represents the canonical PSO, where the particle’s decisions are affected by the particle’s best solution and a swarm’s best solution, with each given the same weighted coefficient.
- *Local and Global Species*: This species is influenced only by its own best and global best position for all swarms.
- *Swarm only Species*: This species is influenced only by the swarm’s best position.
- *Global only species*: This species is influenced only by the global best position for all swarms.
- *Random species*: This species employs randomness, where all three parameters in the PSO velocity function (Eq. (14)) are taken into consideration. For each iteration, the coefficient weights of the parameters are uniformly distributed random number in the range $(0,1)$.

Table 1 shows the parameters for the five species. Once the position of a particle in the swarm has been updated according to equations (14) and (15), it’s fitness value is determined according to equation (12). If the new fitness value is better than the previous one, then the respective best values are updated by the current values.

Algorithm 1 illustrates the Multi-Swarm PSO-based optimization process. The input to the algorithm is the workflow wf and all the available cloud-fog VMs, while the output is the task-VM mapping schedule - \mathbf{gBest} and $F(\mathbf{gBest})$. Parameters N , S and G denote the number of particles,

Algorithm 1 The Proposed Multi-Swarm PSO-Based Algorithm

```

1 Input: Workflow wf composed of  $t$  tasks and all
   available cloud-fog VMs;
2 Output: Task-VM mapping schedule - gBest and
    $F(\mathbf{gBest})$ ;
3 Initialize  $N, S, A, B, C, \omega$ , and  $G$ ;
4 Randomly generate  $N$  particles;
5 Divide the  $N$  particles into  $S$  subswarms of equal sizes;
6  $F(\mathbf{gBest}) \leftarrow 0$ ;
7 for  $s \leftarrow 1, S$  do
8    $F(\mathbf{sBest}^s) \leftarrow 0$ ;
9   for  $i \leftarrow 1, N/S$  do
10    Invoke the Fogworkflowsim workflow
       scheduler;
11    Compute the fitness function value,  $F(\mathbf{x}_i^s)$ , for
       particle  $i$  in swarm  $s$ , by using equation (12) in
       Section IV;
12     $\mathbf{pBest}_i^s \leftarrow \mathbf{x}_i^s$ ;
13     $F(\mathbf{pBest}_i^s) \leftarrow F(\mathbf{x}_i^s)$ ;
14    if  $F(\mathbf{x}_i^s) > F(\mathbf{sBest}^s)$  then
15       $\mathbf{sBest}^s \leftarrow \mathbf{x}_i^s$ ;
16       $F(\mathbf{sBest}^s) \leftarrow F(\mathbf{x}_i^s)$ ;
17    if  $F(\mathbf{x}_i^s) > F(\mathbf{gBest})$  then
18       $\mathbf{gBest} \leftarrow \mathbf{x}_i^s$ ;
19       $F(\mathbf{gBest}) \leftarrow F(\mathbf{x}_i^s)$ ;
20  $k \leftarrow 0$ ;
21 while  $k \leq G$  do
22   for  $s \leftarrow 1, S$  do
23     for  $i \leftarrow 1, N/S$  do
24       (1) Update  $\mathbf{v}_i^s$  and  $\mathbf{x}_i^s$  by using equations (14)
           and (15), and species-specific parameters
           from Table 1;
25       (2) Invoke the Fogworkflowsim workflow
           scheduler;
26       (3) Compute the fitness function value,
            $F(\mathbf{x}_i^s)$ , by using the Weighted Sum
           Objective Function (equation (12) in
           Section IV);
27       if  $F(\mathbf{x}_i^s) > F(\mathbf{pBest}_i^s)$  then
28          $\mathbf{pBest}_i^s \leftarrow \mathbf{x}_i^s$ ;
29          $F(\mathbf{pBest}_i^s) \leftarrow F(\mathbf{x}_i^s)$ ;
30       if  $F(\mathbf{x}_i^s) > F(\mathbf{sBest}^s)$  then
31          $\mathbf{sBest}^s \leftarrow \mathbf{x}_i^s$ ;
32          $F(\mathbf{sBest}^s) \leftarrow F(\mathbf{x}_i^s)$ ;
33       if  $F(\mathbf{x}_i^s) > F(\mathbf{gBest})$  then
34          $\mathbf{gBest} \leftarrow \mathbf{x}_i^s$ ;
35          $F(\mathbf{gBest}) \leftarrow F(\mathbf{x}_i^s)$ ;
36    $k \leftarrow k + 1$ ;

```

the number of swarms, and the number of generations respectively, while the other parameters have already been defined in Section IV.

The algorithm starts with the initialization of the values of the following parameters: N, S, A, B, C, ω , and G . In the next step, N particles are created and divided into S swarms of equal sizes and the fitness value of the best global position $F(\mathbf{gBest})$ is initialized to 0.

Between line 7 and line 19, the fitness values of the particles in each swarm are determined according to equation (12), by running the respective workflow scheduling simulation on Fogworkflowsim [33]. The $\mathbf{pBest}_i^s, \mathbf{sBest}^s$ and \mathbf{gBest} values are determined among all the swarms. Once this process is completed, each of the swarms/species is evolved by swarm/species-specific parameter settings presented in Table 1 in an iterative process until G generations are reached. This happens from 21 to line 36. At every iteration, the computation goes through two loops: 1) the outer loop for the swarms; 2) the inner loop for the individual particles in a particular swarm. Within the inner loop, there are three steps:

- 1) The update of the particle's velocity and position by using equations (14) and (15), and species-specific parameters from Table 1.
- 2) The invocation of the Fogworkflowsim workflow scheduler to determine the fitness of the particle determined in the first step.
- 3) The calculation of the fitness function value, $F(\mathbf{x}_i^s)$, by using the Weighted Sum Objective Function (equation (12), in Section IV).

From line 27 to line 35, the newly determined fitness value $F(\mathbf{x}_i^s)$ for a particle i in swarm s is compared with the three best fitness values associated with the particle at personal, swarm, and global levels $F(\mathbf{pBest}_i^s), F(\mathbf{sBest}^s)$ and $F(\mathbf{gBest})$ respectively). The values for $\mathbf{pBest}_i^s, F(\mathbf{pBest}_i^s), \mathbf{sBest}^s, F(\mathbf{sBest}^s), \mathbf{gBest}$, and $F(\mathbf{gBest})$ are updated, whenever better values are found in each case.

VI. PERFORMANCE EVALUATION

This section begins by describing the workflow models along with the simulation environment setup on the FogWorkflowSim Toolkit [33]. These preliminaries are followed by the experimental results and discussion.

A. WORKFLOW MODELS

This study uses five well-known scientific workflows [39] from various scientific domains for effective evaluation of the proposed algorithm. These scientific workflows are published by the Pegasus project [40], where the DAG XML file representations for the respective workflows are used as input for the simulations. These workflows are composed of a number of tasks, dependencies, run-times and data required to be transferred between different tasks. Fig. 4 shows a simplified graphical structure of the workflows. These workflows are described as follows:

- 1) *Montage workflow*: This represents an astronomy application that creates custom mosaics of the sky with multiple input images.
- 2) *CyberShake workflow*: This is used to characterize earthquake hazards threatening a region.

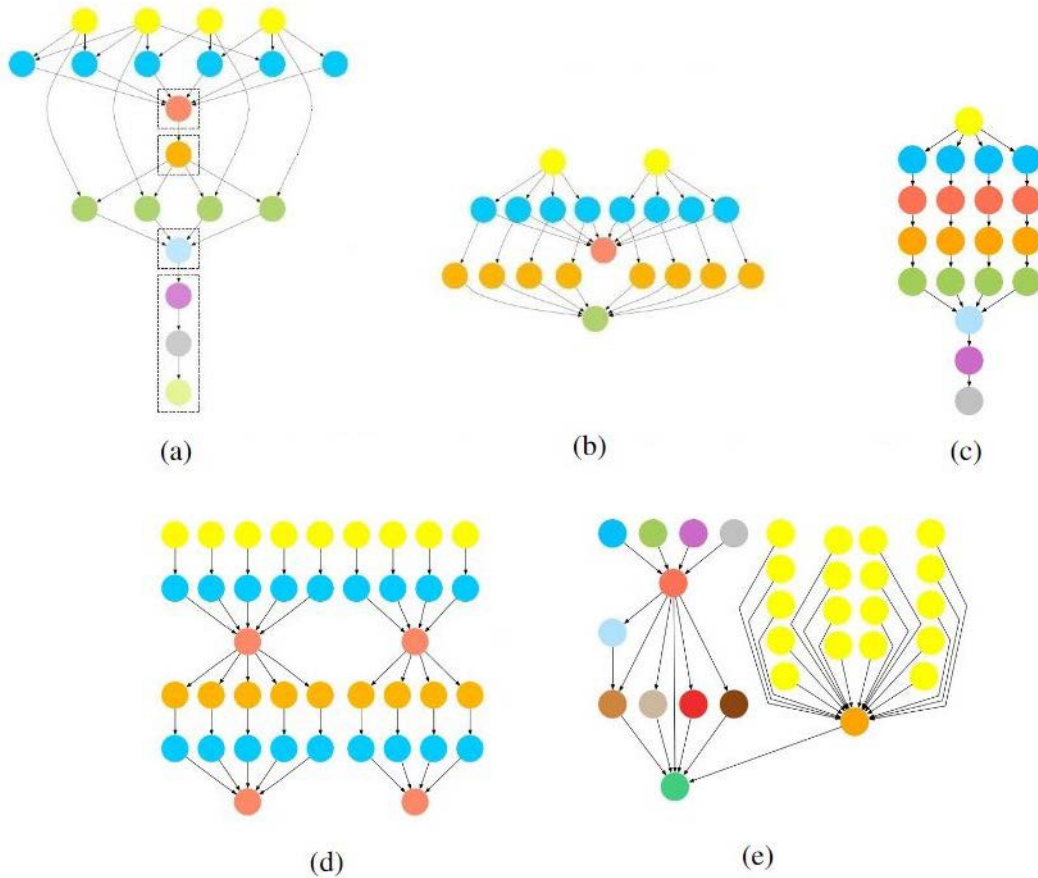


FIGURE 4. Structure of scientific workflows: (a) Montage, (b) CyberShake, (c) Epigenomics, (d) LIGO and (e) SIPHT.

- 3) *Epigenomics workflow*: This is used in the field of bioinformatics to automate the different operations in genome sequence processing.
- 4) *LIGO Inspirial Analysis workflow*: This is for the detection of gravitational waves
- 5) *Sipht workflow*: This is for automating the search for sRNA encoding-genes of all of the bacterial replications.

TABLE 2. Parameter settings of cloud-fog environment.

Parameters	End Device	Fog VM	Cloud VM
Processing rate (MIPS)	500	1000	2000
Task execution cost (\$)	0	0.48	0.96
Communication cost (\$)	0	0.01	0.02
Working power (mW)	200	700	1700
Idle power (mW)	50	200	1200
Uplink bandwidth (Mbps)	800	500	300
Downlink bandwidth (Mbps)	1000	800	500

B. SIMULATION ENVIRONMENT

The overall setup of the experiments and the simulation environment is described in this subsection. The simulations are conducted using FogWorkflowSim [33], which is executed on a computer with 64-bit Windows 10 operating system, Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz and 16 GB RAM. The performance of the proposed MS-PSO is evaluated on five scientific workflow types, as described previously and the results are compared with four other population-based algorithms, namely, standard GA, PSO, DE and GA-PSO proposed in [14]. The final parameter settings for all the experiments are based on preliminary runs. The population size = 50 for each algorithm.

The MS-PSO learning factors *A*, *B* and *C* are set according to Table 1, with *S* = 5 and each containing 10 particles. The inertia weight $\omega = 1$. The classical PSO learning factors $C_1 = C_2 = 2$, with inertia weight $\omega = 1$. The GA algorithm’s crossover and mutation rates are 0.8 and 0.1, respectively. The DE crossover probability = 0.9 and the differential weight = 0.8. The number of iterations for all algorithms is 100. The simulations are executed 10 times for each workflow and task volume to get the average performance of the algorithms. The simulator is setup with one end device, six fog VMs and ten cloud VMs. The characteristics for each VM on the three cloud-fog layers along with the environment settings are shown in Table 2.

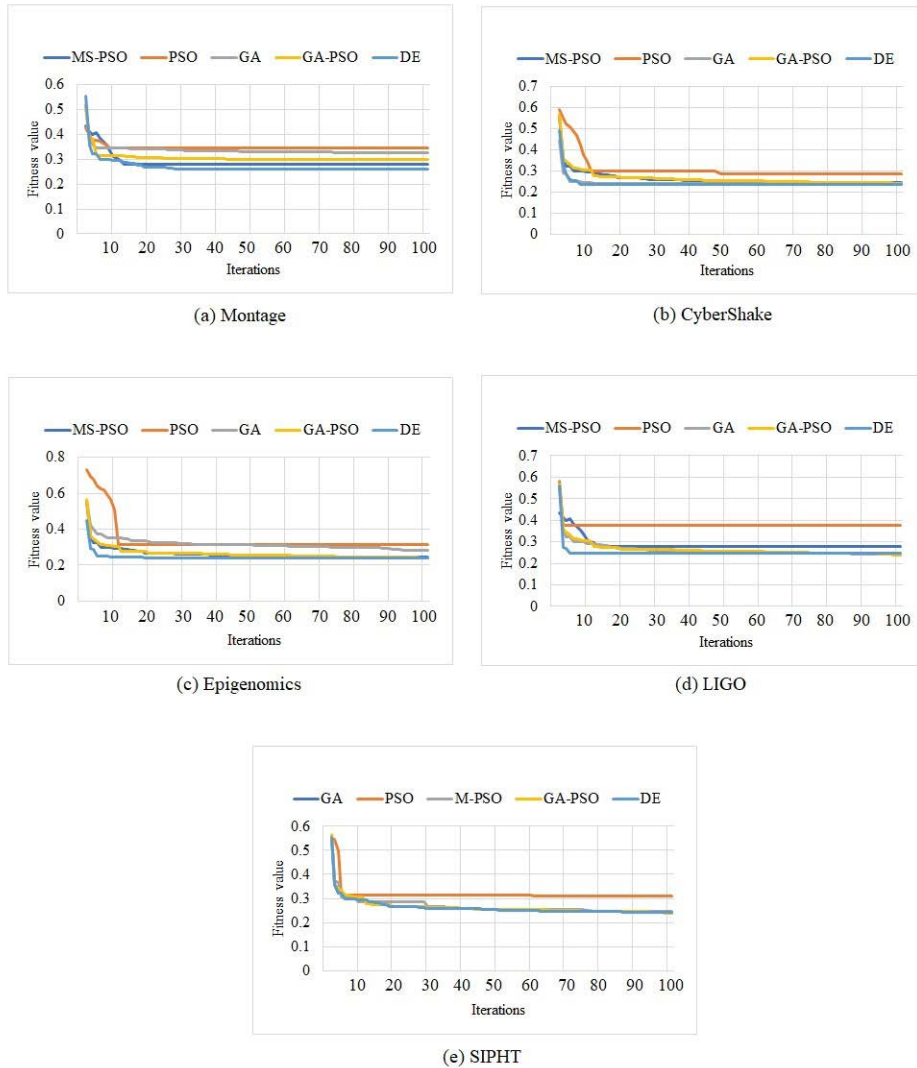


FIGURE 5. Comparison of fitness value evolution.

C. SIMULATION RESULTS

The performance of the proposed algorithm is compared with the canonical PSO and the GA, DE and GA-PSO, which are the three other population-search based algorithms for cloud based workflow scheduling. All the algorithms are executed on the same simulation environment setup. The respective fitness evolution graphs and run-times for each algorithm are recorded and presented. Finally, the comparison of the results is presented in terms of the following performance metrics: makespan, cost, energy and load balancing on the cloud and fog respectively.

Fig. 5 shows the evolution of the fitness value for each scientific workflow with 300 tasks and the overall quality of the task schedule produced by the algorithms considering all the performance metrics. For the Montage workflow, as shown in Fig. 5(a), the MS-PSO approach closely follows the DE approach, which achieves the best

fitness value. On the other hand the canonical PSO approach exhibits the worst performance closely followed by the GA approach. In the case of Cybershake (Fig. 5(b)), the rest of the algorithms apart from the PSO approach achieve the similar fitness value. The GA and the DE approaches seem to converge faster than the rest of the algorithms. The Epigenomics workflow, in Fig. 5(c), exhibits a similar pattern to the Montage workflow (see Fig. 5(a)) in the sense that MS-PSO and DE achieve the lowest fitness values, while GA and PSO achieves the highest values. In Fig. 5(d), the MS-PSO again follows the DE, GA-PSO and GA algorithms, which are the best performing approaches, closely. The SIPHT workflow, Fig. 5(e), shows a similar pattern. In all these cases, all algorithms seem to converge by the 30th iteration.

From the results in Fig. 5, it can clearly be seen that the final fitness value of the proposed MS-PSO is much

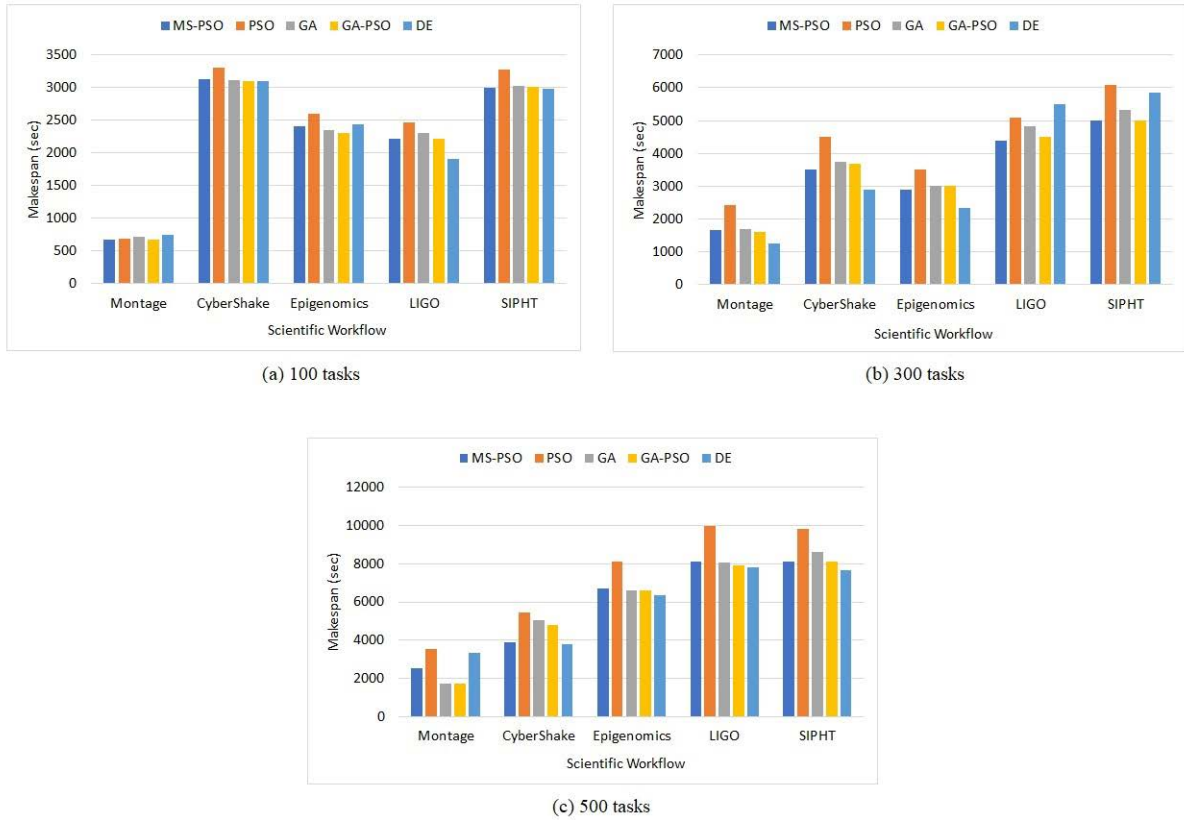


FIGURE 6. Comparison of makespan for the different algorithms.

smaller than the canonical PSO and very similar to the other algorithms. This indicates that the MS-PSO has the ability to find near-optimal solutions, while the canonical PSO easily falls into a local optima. The MS-PSO benefits from the multi-species approach where the particle’s velocity update in each swarm is calculated differently, enabling a more exploratory global search with the capability to easily jump local optima.

Fig. 6 - Fig. 10 shows the average performance metric results of the task schedules realized by the five algorithms for five workflow types under 100, 300, and 500 tasks. Fig. 6 shows that the Montage workflow achieves the shortest makespan of the five workflows. At 100 tasks, all the algorithms exhibit a similar performance. At 300 tasks, PSO is the worst while DE is the best approach. The rest of the algorithms lie in between. At 500 tasks, DE joins PSO as the worst performing algorithms, while GA and GA-PSO achieve the best performance. The MS-PSO shows stability and reliability.

In the Cybershake workflow, the PSO algorithm is the worst at 100 and 300 tasks, and it is joined by PSO, GA-PSO and GA at 500 tasks. At the other end, the rest of the algorithms perform equally at 100 tasks. The DE is the best at 300 tasks and it is joined by the MS-PSO at 500 tasks. Again, the proposed MS-PSO exhibits a better performance

than the canonical PSO and it is more stable than the rest of the algorithms. In the Epigenomics workflow, performance is more or less the same at 100 tasks. At 300 tasks, PSO is the worst, while DE is the best. At 500 tasks, PSO is still the worst while the rest of the algorithms perform more or less in a similar manner.

In the LIGO workflow, DE is the best, while the rest of the approaches exhibit similar performance, at 100 tasks. At 300 tasks, the DE is the worst, while the MS-PSO is the best. At 500 tasks, PSO is the worst, while the rest of the algorithms exhibit similar performance. Finally, in the SIPHT workflow, PSO has worst makespan at 100 tasks, while the rest of the algorithms exhibit a similar performance. At 300 tasks, PSO is joined by DE with the worst makespan. The MS-PSO and GA-PSO achieve the best makespan values. At 500 tasks, PSO still performs poorly, while the rest of the algorithms perform similarly.

The proposed MS-PSO approach clearly outperforms the canonical PSO approach, while competing fairly well with the other approaches. It comes out on top in several instances, while performing closer to the top in many cases. Amongst the competitors, DE and GA-PSO produce the best results on several instances, but they also register the worst results in a number of cases. The proposed MS-PSO approach is, therefore, stable and more reliable. This can be attributed

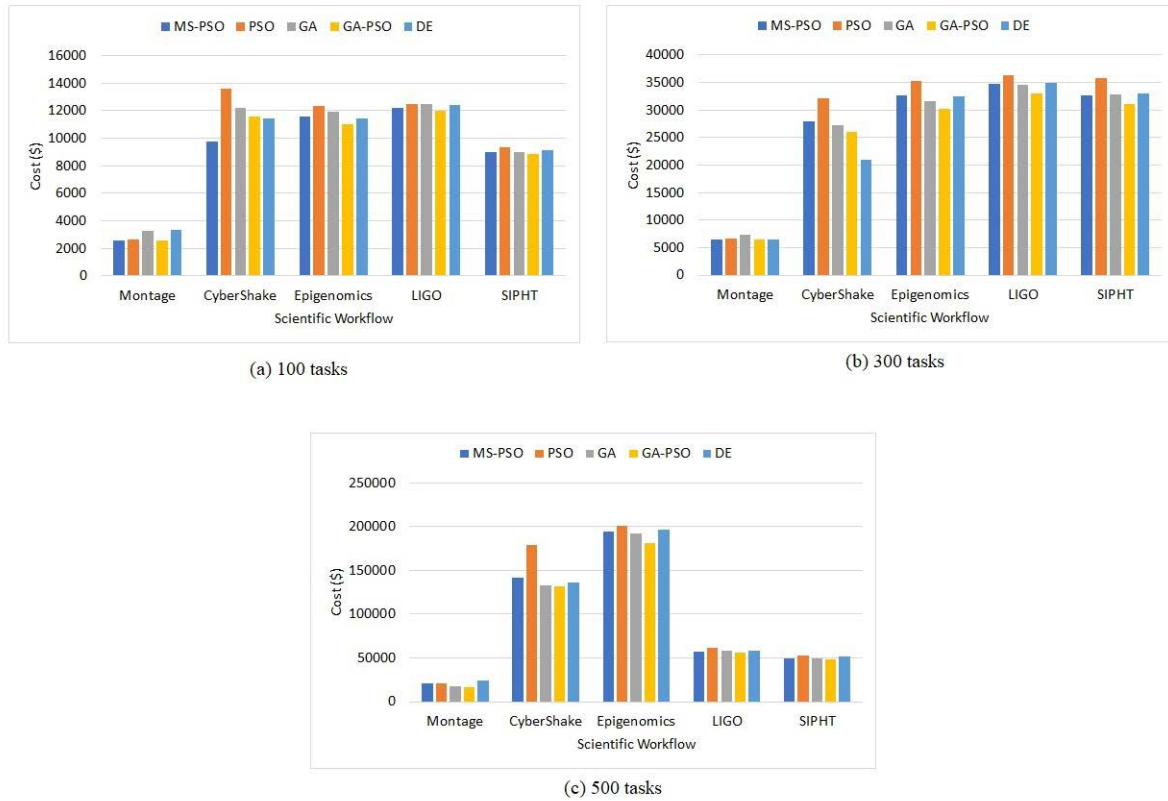


FIGURE 7. Comparison of cost for the different algorithms.

to the use of different velocity update mechanisms in the sub-swarms. This helps to guarantee efficiency in times of when some sub-swarms become unstable or are stuck in the local minima. The MS-PSO generally achieves better results than the GA-PSO. This suggests that the parallel approach of combining algorithms might probably generate good results, as opposed to the serial approach in the GA-PSO approach.

Fig. 7 illustrates a comparative evaluation based on cost. The Montage workflow yields the lowest makespan and as a result, it also registers the lowest cost. At 100 tasks, MS-PSO, PSO and GA-PSO achieve the lowest cost, while GA and DE are the worst in terms of cost. There is, however, similar performance among the algorithms when the tasks are increased to 300 and 500. For the Cybershake workflow, MS-PSO has the lowest cost at 100 tasks, while PSO has the highest cost. When the number of tasks is increased to 300, DE yields the lowest cost, while PSO still has the highest cost. The proposed MS-PSO ranks between the two extremes. At 500 tasks, PSO remains with the highest cost, while the rest of the algorithms have similar cost values. For the Epigenomics workflow, the performance is more or less the same for all algorithms for all tasks. The same applies to the LIGO workflow. For the SIPHT workflow, all algorithms register similar costs for 100 and 500 tasks. For 300 tasks, only the PSO yields a high cost while the rest of the algorithms have

similar costs. On the overall cost analysis, the PSO gives the highest cost, while the proposed MS-PSO still competes fairly well with the rest of the algorithms. MS-PSO and DE come out on top once.

Fig. 8 illustrates a comparative evaluation based on energy consumption. As expected, the Montage workflow consumes the lowest amount of energy under all tasks. At 100 tasks, all algorithms have similar energy consumption, while PSO and the DE are the worst and the best performing algorithms when the number of tasks is increased to 300. At 500 tasks, the DE becomes the worst algorithm while GA and GA-PSO are the best; MS-PSO and PSO exhibit similar performance. For the Cybershake workflow, MS-PSO has the lowest energy consumption at 100 tasks, while the rest of the algorithms exhibit similar performance. At 300 tasks, DE yields the lowest energy consumption, while PSO still has the highest energy consumption. The proposed MS-PSO ranks between the two extremes. At 500 tasks, MS-PSO gives the best performance, closely followed by DE. The rest of the algorithms have similar energy values.

For the Epigenomics workflow, PSO consumes the highest amount of energy while the rest of the algorithms show similar energy consumption for 100 and 500 tasks. At 300 tasks, the trend is the same, but DE yields slightly better performance than the other algorithms. For the LIGO

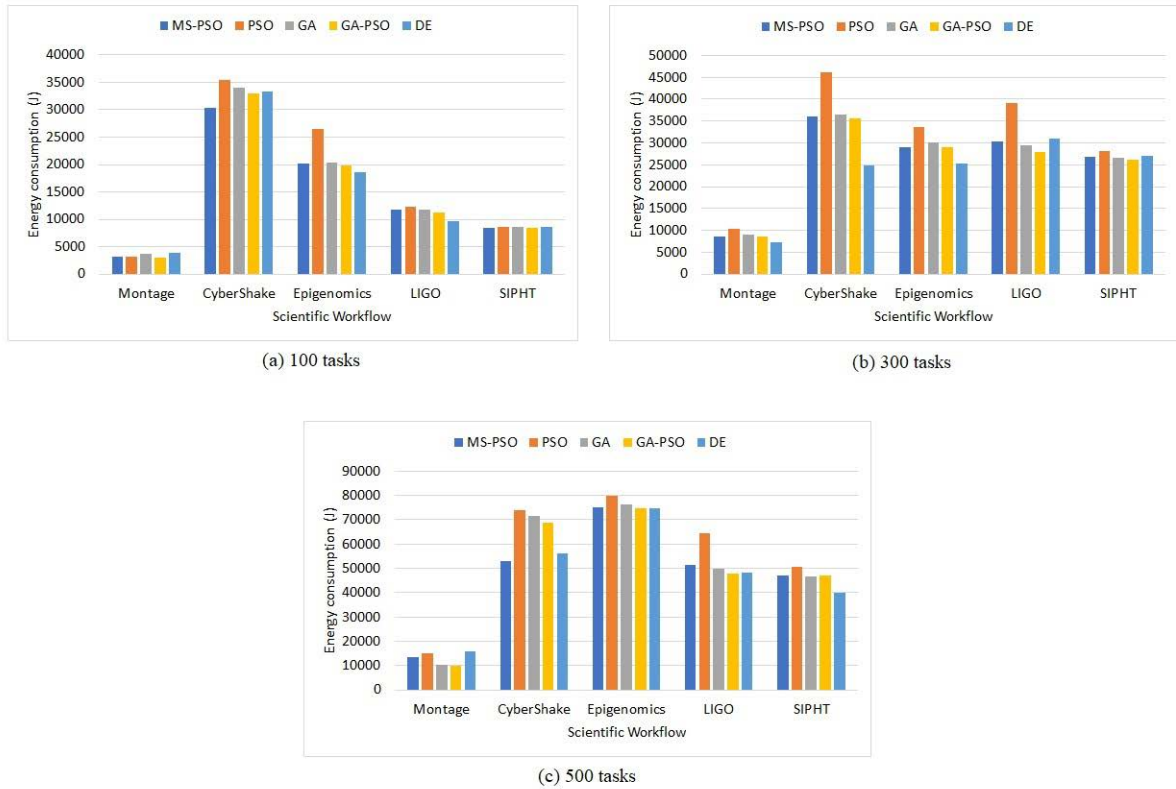


FIGURE 8. Comparison of energy consumption for the different algorithms.

workflow, DE slightly outperforms the rest of the algorithms for 100 tasks. On the other hand, PSO consumes the highest energy, while the rest of the algorithms yield similar energy consumption values for 300 and 500 tasks. For the SIPHT workflow, all algorithms register similar costs for 100 and 500 tasks. For 300 tasks, only the PSO yields the highest energy consumption while DE has a slightly lower energy consumption than the other three competitors.

On the overall energy analysis, the proposed MS-PSO and DE yield the lowest energy consumption on several instances. Ironically, DE also registers the highest energy consumption on one instance. This further confirms the fact that the DE algorithm is unstable and unreliable. This stems from the fact that it uses one homogeneous population. Therefore, the possibility of premature convergence is always there.

Fig. 9 illustrates a comparative evaluation based on load balancing on the cloud resources. For the Montage workflow, all algorithms register similar load balancing performance for 100 tasks. At 300 tasks, PSO and the DE are the worst and the best performing algorithms, respectively. On the other hand, GA performs best at 500 tasks, with PSO, GA-PSO and DE exhibiting the worst performance. MS-PSO’s load balancing performance is in between the two extremes. For the Cybershake workflow, MS-PSO and PSO have the best and worst performance for 100 tasks,

respectively. At 300 tasks, DE and PSO have the best and worst performance, while at 500 tasks DE and MS-PSO have the best performance. For the Epigenomics workflow, PSO has the worst performance while the rest of the algorithms show similar performance for 100 tasks. The DE yields better performance than the other algorithms for 300 and 500 tasks. For the LIGO workflow, all algorithms except PSO, exhibit similar performance for all tasks. For the SIPHT workflow, PSO and MS-PSO have the worst performance while the rest of the algorithms register similar costs for 100 tasks. For 300 tasks, PSO and DE have the best and worst performance, while at 500 tasks, only PSO performs badly.

In the overall analysis on load balancing on the cloud, DE yields the best performance on most instances while MS-PSO ranks second. Nevertheless DE yields the worst performance on one instance. This further confirms the problem of instability as previously explained.

Fig. 10 illustrates a comparative evaluation based on load balancing on the fog resources. For the Montage workflow, when the number of workflow tasks is set to 100, MS-PSO achieves the best load balancing performance, with the GA giving the worst performance. PSO performs slightly poorly, while the rest of the algorithms perform similarly for 300 and 500 tasks. For the Cybershake workflow, DE and PSO have the best and worst performance for 100 tasks. At 300 tasks, all algorithms have similar performance, while DE and GA

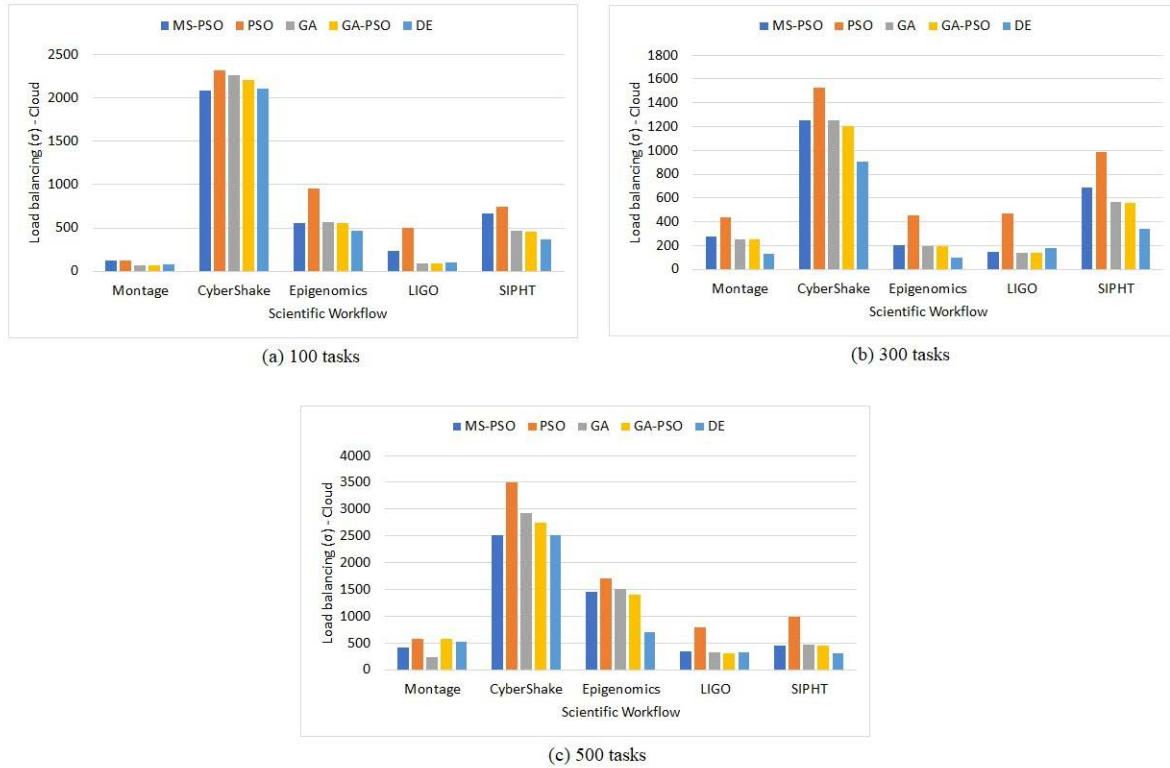


FIGURE 9. Comparison of load balancing on the cloud for the different algorithms.

have the best and worst performance for 500 tasks. For the Epigenomics workflow, GA-PSO and PSO have the best and worst performance for 100 and 500 tasks. On the other hand, DE and PSO have the best and worst performance for 300 tasks. For the LIGO workflow, GA-PSO and PSO have the best and worst performance for 100 tasks. At 300 tasks, DE slightly outperforms GA-PSO to emerge as the best performing algorithm, while only PSO performs badly at 500 tasks. For the SIPHT workflow, PSO has the worst performance; the MS-PSO follows PSO closely, while the rest of the algorithms register similar costs for 100 tasks. For 300 tasks, PSO and DE have the best and worst performance, while at 500 tasks, just like in the case of cloud-based load balancing, only PSO performs badly.

In the overall analysis on load balancing on the fog devices, DE yields the best performance on most instances while GA-PSO and MS-PSO rank second and third respectively.

Having compared the algorithms based on various performance metrics, it is also important to consider the computational times of the algorithms. Therefore, Table 3 shows a comparison of the average execution times of the algorithms for the five scientific workflows using different number of tasks. GA and DE have the worst execution times. On the other hand, the approaches that incorporate PSO have much better execution times, with the canonical PSO clearly outperforming the rest of the algorithms in all

cases. This highlights the computational and implementation simplicity of the canonical PSO technique. The proposed MS-PSO algorithm inherits those attributes, albeit at a slightly increased computational cost due to the addition of the third term that captures the contribution of the best position for each swarm and determining each swarms' best performing particle for every iteration. However, the canonical PSO produced the lowest quality of solutions as shown in Fig. 5 - Fig. 10. It is noteworthy that as the workflow size increases, the MS-PSO offers a good trade-off between algorithm execution time and quality of solutions produced. This is crucial especially for large-scale scientific workflow applications where long running times can have a significant impact on financial cost and energy when generating task schedules for a cloud-fog environment.

These results show that the canonical PSO algorithm performs the worst on all the workflow instances due to PSO's well-known problem of premature convergence, which leads it to getting trapped in a local optima. The MS-PSO algorithm, on the other hand, outperforms the canonical PSO for the five scientific workflows and competes fairly well with the other algorithms. The MS-PSO also exhibits stability; it has rarely offered the worst performance. This demonstrates the high reliability of the MS-PSO approach. The MS-PSO benefits from the variety of cognitive and social learning coefficients of the different swarms when

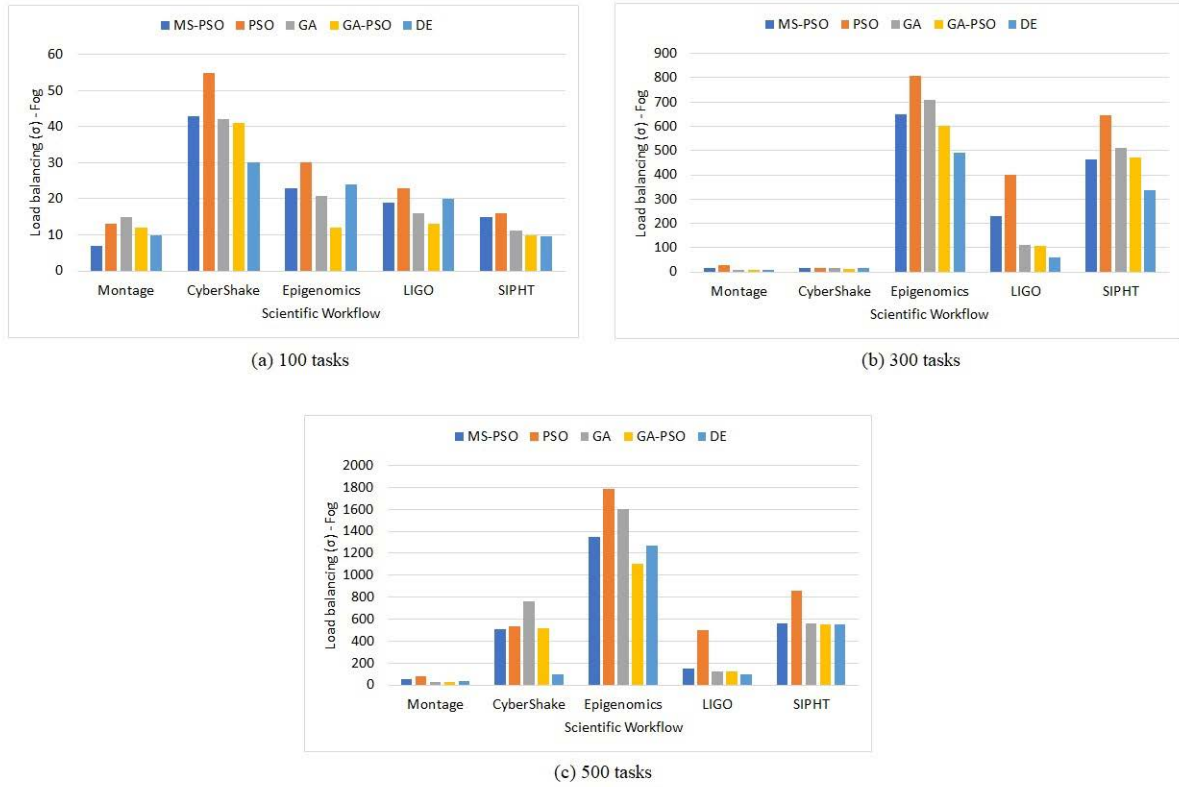


FIGURE 10. Comparison of load balancing on the fog for the different algorithms.

TABLE 3. The average execution time of the algorithms.

Workflow	Number of tasks	Algorithm runtime /sec				
		MS-PSO	PSO	GA	GA-PSO	DE
Montage	100	14.961	10.656	25.247	16.523	25.424
	300	56.456	51.279	76.009	60.241	77.302
	500	158.644	153.408	192.513	165.19	194.139
CyberShake	100	14.422	10.395	24.174	15.908	24.266
	300	48.333	42.252	61.028	49.897	69.346
	500	130.358	115.924	145.265	133.29	144.698
Epigenomics	100	13.061	10.819	24.107	15.344	24.37
	300	56.405	51.78	71.793	58.782	76.651
	500	178.086	171.654	209.732	184.292	207.713
LIGO	100	13.233	10.407	35.116	17.892	36.117
	300	59.88	55.063	76.776	64.797	79.474
	500	170.164	161.873	196.134	174.329	201.607
SIPHT	100	109.573	104.274	125.372	112.439	120.545
	300	368.48	362.949	392.557	373.915	390.109
	500	740.411	730.311	777.945	747.912	782.624

determining the velocity for each particle, hence the algorithm is able to escape local optima. The results are affected by the characteristics of the workflow instances. When the number of tasks for the respective workflow instance is higher, the metric values are also generally higher as the run times and data sizes of the workflow DAG are significantly higher.

VII. CONCLUSION

This work has proposed a Multi-Swarm Particle Swarm Optimisation (MS-PSO) approach for scientific workflow scheduling in cloud-fog environments. This approach is motivated by the problem of premature convergence in the PSO technique, an algorithm that has largely been used for scientific workflow scheduling due to its simplicity and

ease of implementation. This work has also developed the concept of load balancing, both for the fog and the cloud resources, and incorporated it in the weighted multi-objective mechanism, initially presented in [32]. The state-of-the-art multi-swarm PSO algorithms have also been reviewed with the aim of determining the best approach for implementing a multi-swarm PSO technique for scientific workflow scheduling. The socio-cognitively inspired PSO [22] has, therefore, been adopted and adapted for this work. The resulting MS-PSO uses multiple swarms of different species of particles to improve the exploration of the search space and avoid premature convergence. Finally, simulations have been performed with five scientific workflows using the FogWorkflowSim Toolkit [33]. The proposed algorithm has been compared with the canonical PSO, GA, DE and GA-PSO based on the following scientific workflows: Montage, Cybershake, Epigenomics, LIGO and SIPHT. The following performance metrics were used: makespan, cost, energy consumption, and load balancing on the fog as well as on the cloud resources.

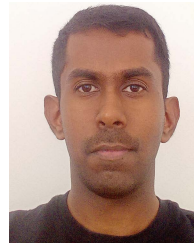
Results have shown that the proposed MS-PSO generally outperforms the canonical PSO on all scientific workflows and under all performance metrics. The proposed MS-PSO generally performs better than GA and GA-PSO. It competes fairly well against DE and more importantly it is more stable and reliable than DE. In terms of computational complexity, the proposed MS-PSO only ranks second to PSO, while DE has the worst execution time. These results show that the proposed MS-PSO technique is exhibiting far much better overall performance compared to the competitors.

As part of future work, multiple species that utilize population update mechanisms from completely different algorithmic frameworks such as DE, MS-PSO, and GA will be explored for scientific workflow scheduling. Furthermore, a framework that hybridizes the proposed multi-swarm PSO-based scheduling mechanism with concepts drawn from dynamic scientific workflow scheduling [41] and multi-objective reinforcement learning-based scientific workflow scheduling [42] will be investigated based on the cloud-fog environment and the four objectives, as proposed in this work.

REFERENCES

- [1] B. Ludäscher, "What makes scientific workflows scientific?" in *Scientific and Statistical Database Management*, M. Winslett, Ed. Berlin, Germany: Springer, 2009, p. 217.
- [2] Z. Ahmad, A. I. Jehangiri, M. A. Alaanzay, M. Othman, R. Latip, S. K. U. Zaman, and A. I. Umar, "Scientific workflows management and scheduling in cloud computing: Taxonomy, prospects, and challenges," *IEEE Access*, vol. 9, pp. 53491–53508, 2021.
- [3] M. A. Rodriguez and R. Buyya, "A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 8, p. e4041, Apr. 2017.
- [4] B. Ludäscher, M. Weske, T. McPhillips, and S. Bowers, "Scientific workflows: Business as usual?" in *Proc. Int. Conf. Bus. Process Manage.* Berlin, Germany: Springer, 2009, pp. 31–47.
- [5] M. A. Miller, W. Pfeiffer, and T. Schwartz, "The CIPRES science gateway: A community resource for phylogenetic analyses," in *Proc. TeraGrid Conf., Extreme Digit. Discovery*. ACM, 2011, pp. 1–8.
- [6] S. Jha, S. Lathrop, J. Nabrzyski, and L. Ramakrishnan, "Incorporating scientific workflows in computing research processes," *Comput. Sci. Eng.*, vol. 21, no. 4, pp. 4–6, Jul. 2019.
- [7] J. Yu and R. Buyya, "A taxonomy of workflow management systems for grid computing," *J. Grid Comput.*, vol. 3, nos. 3–4, pp. 171–200, Sep. 2005.
- [8] R. Duan, R. Prodan, and X. Li, "Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 29–42, Jan./Mar. 2014.
- [9] Y. Zhao, Y. Li, I. Raicu, S. Lu, C. Lin, Y. Zhang, W. Tian, and R. Xue, "A service framework for scientific workflow management in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 6, pp. 930–944, Nov. 2015.
- [10] W. Song, F. Chen, H. 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] R. Buyya, S. Pandey, and C. Vecchiola, "Cloudbus toolkit for market-oriented cloud computing," in *Proc. 1st Int. Conf. Cloud Comput.*, vol. 5931, 2009, pp. 24–44.
- [12] A. Verma and S. Kaushal, "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling," *Parallel Comput.*, vol. 62, pp. 1–19, Feb. 2017.
- [13] Y. Kothiyari and A. Singh, "A multi-objective workflow scheduling algorithm for cloud environment," in *Proc. 3rd Int. Conf. Internet Things, Smart Innov. Usages (IoT-SIU)*, Feb. 2018, pp. 1–6.
- [14] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–16, Jan. 2018.
- [15] M. Farid, R. Latip, M. Hussin, and N. A. W. A. Hamid, "A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, p. 551, Apr. 2020.
- [16] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of pso-based scheduling algorithms in cloud computing," *J. Neww. Syst. Manage.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.
- [17] J. Kennedy and R. Eberhart, "Particle swarm optimization in: Neural networks," in *Proc. IEEE Int. Conf.*, Nov. 1992, pp. 1942–1948.
- [18] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud-edge environment," *Future Gener. Comput. Syst.*, vol. 97, pp. 361–378, Aug. 2019.
- [19] B. Wei, Y. Tang, X. Jin, M. Jiang, Z. Ding, and Y. Huang, "A dynamic multi-swarm particle swarm optimization with global detection mechanism," *Int. J. Cognit. Informat. Natural Intell.*, vol. 15, no. 4, pp. 1–23, Oct. 2021.
- [20] J. Lu, J. Zhang, and J. Sheng, "Enhanced multi-swarm cooperative particle swarm optimizer," *Swarm Evol. Comput.*, vol. 69, Mar. 2022, Art. no. 100989.
- [21] J.-J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proc. IEEE Swarm Intell. Symp.*, Jun. 2005, pp. 124–129.
- [22] I. Bugajski, P. Listkiewicz, A. Byrski, M. Kisiel-Dorohinicki, W. Korczynski, T. Lenaerts, D. Samson, B. Indurkha, and A. Nowé, "Enhancing particle swarm optimization with socio-cognitive inspirations," *Proc. Comput. Sci.*, vol. 80, pp. 804–813, Jan. 2016.
- [23] S. Wang, G. Liu, M. Gao, S. Cao, A. Guo, and J. Wang, "Heterogeneous comprehensive learning and dynamic multi-swarm particle swarm optimizer with two mutation operators," *Inf. Sci.*, vol. 540, pp. 175–201, Nov. 2020.
- [24] F. T. Varna and P. Husbands, "HIDMS-PSO: A new heterogeneous improved dynamic multi-swarm PSO algorithm," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2020, pp. 473–480.
- [25] X. Xia, Y. Tang, B. Wei, Y. Zhang, L. Gui, and X. Li, "Dynamic multi-swarm global particle swarm optimization," *Computing*, vol. 102, no. 7, pp. 1587–1626, Jul. 2020.
- [26] X. Jiang, Y. Yue, Y. Min, Q. Zhang, and X. Gui, "Particle swarm optimization with multiple adaptive subswarms," *J. Phys., Conf.*, vol. 1757, no. 1, Jan. 2021, Art. no. 012024.
- [27] J. Guzmán, M. García-Valdez, and J. J. Merelo-Guervós, "Can communication topology improve a multi-swarm PSO algorithms?" in *Proc. Workshop Eng. Appl. Cham, Switzerland: Springer*, 2021, pp. 3–12.
- [28] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. IEEE 8th Int. Conf. E-Science*, Oct. 2012, pp. 1–8.

- [29] H. F. Atlam, R. J. Walters, and G. B. Wills, "Fog computing and the Internet of Things: A review," *Big Data Cogn. Comput.*, vol. 2, no. 2, p. 10, 2018.
- [30] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *J. Syst. Archit.*, vol. 98, pp. 289–330, Sep. 2019.
- [31] V. De Maio and D. Kimovski, "Multi-objective scheduling of extreme data scientific workflows in fog," *Future Gener. Comput. Syst.*, vol. 106, pp. 171–184, May 2020.
- [32] D. Subramoney and C. N. Nyirenda, "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Dec. 2020, pp. 760–767.
- [33] X. Liu, L. Fan, J. Xu, X. Li, L. Gong, J. Grundy, and Y. Yang, "FogWorkflowSim: An automated simulation toolkit for workflow performance evaluation in fog computing," in *Proc. 34th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Nov. 2019, pp. 1114–1117.
- [34] F. Wang, H. Zhang, and A. Zhou, "A particle swarm optimization algorithm for mixed-variable optimization problems," *Swarm Evol. Comput.*, vol. 60, Feb. 2021, Art. no. 100808.
- [35] H. J. Na and S. Yoo, "PSO-based dynamic UAV positioning algorithm for sensing information acquisition in wireless sensor networks," *IEEE Access*, vol. 7, pp. 77499–77513, 2019.
- [36] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
- [37] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.
- [38] S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *Sci. World J.*, vol. 2013, pp. 1–13, Sep. 2013.
- [39] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Proc. 3rd Workshop Workflows Support Large-Scale Sci.*, Nov. 2008, pp. 1–10.
- [40] *The Pegasus Website*. Accessed: Jan. 10, 2022. [Online]. Available: <https://pegasus.isi.edu/>
- [41] J. Sahni and D. P. Vidyarthi, "A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment," *IEEE Trans. Cloud Comput.*, vol. 6, no. 1, pp. 2–18, Mar. 2015.
- [42] Y. Qin, H. Wang, S. Yi, X. Li, and L. Zhai, "A multi-objective reinforcement learning algorithm for deadline constrained scientific workflow scheduling in clouds," *Frontiers Comput. Sci.*, vol. 15, no. 5, pp. 1–12, Oct. 2021.



DINESHAN SUBRAMONEY received the bachelor's degree (Hons.) in computer science from the University of KwaZulu-Natal, South Africa, in 2014. He is currently pursuing the M.Sc. degree in computer science with the University of the Western Cape, South Africa. He has been working in the IT industry, since 2015. His research interests include workflow scheduling, cloud computing, and fog computing.



CLEMENT N. NYIRENDA received the B.Sc. degree in electrical engineering from the University of Malawi, in 2000, the M.Sc. degree in computer engineering from the University of KwaZulu-Natal, South Africa, in 2007, and the Ph.D. degree in computational intelligence from the Tokyo Institute of Technology, Japan, in 2017. From 2011 to 2012, he was a Specially Appointed Assistant Professor at Keio University, Japan. From 2012 to 2018, he was a Lecturer/a Senior Lecturer at University Namibia. Since 2018, he has been a Senior Lecturer in computer science at the University of the Western Cape. He has published more than 50 articles in computational intelligence paradigms, such as fuzzy systems, evolutionary computation and artificial neural networks and their applications in communication networks, and smart environments.

He was a recipient of the *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)* Young Researcher Award, in 2013.

• • •