INTERNATIONAL JOURNAL OF Telecommunications AIR DEFENSE COLLEGE

Enhanced Particle Swarm Optimization for Task Offloading and Scheduling in Cloud-Fog Environment

Citation: Gamal, M.; Awad, S.; Abdel-Kader, R; Khaled Abd El Salam, K. *IJT'2024, Vol. 04, Issue 02, pp. 01-16, 2024.*

Editor-in-Chief: Youssef Fayed.

Received: 29/07/2024.

Accepted: 11/10/2024.
Published: 11/10/2024.

Marwa Gamal[1], Samar Awad[1,*], Rehab F. Abdel-Kader[2], and Khaled Abd El Salam[1,3]

[1]Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt.
[2]Electrical Engineering Department, Faculty of Engineering, Port Said University, Port Said, Egypt.
[3]Department of Information System, College of Information Technology, Misr University for Science & Technology (MUST), 6th of October City 12566, Egypt.
* Correspondence: Samar_Awad@eng.suez.edu.eg; Tel.: (01204016520)

Abstract: The explosion of devices and their varied uses in the Internet of Things (IoT) have created a massive quantity of data that requires significant processing power. Fog computing, as a prolongation of cloud computing, presents a promising new model by bringing processing power closer to users through fog servers. Compared to accessing distant cloud servers, this significantly reduces latency, or the required time for data to travel. This setup allows users to offload tasks to nearby servers, ultimately improving the Quality of Service (QoS) they experience. Finding the best match between workflow tasks and available resources is critical to minimizing completion time (makespan), especially in delay-sensitive applications requiring fast data processing. However, achieving this optimal match remains a challenge. This work proposes an Enhanced Particle Swarm Optimization (EPSO) algorithm specifically designed to address this challenge. The performance of EPSO is compared against PSO, Max-Min, and Round-Robin (RR) scheduling methods. Simulations are conducted using diverse scientific workflow domains. The results demonstrate that EPSO outperforms all other methods in minimizing makespan across all tested workflows. Furthermore, EPSO exhibits competitive performance in other metrics like energy consumption and cost while maintaining greater stability and reliability.

Keywords: Cloud-Fog Computing; Offloading; Scientific Workflow; PSO; Task Scheduling.

1. Introduction

The telecommunication networks' growth is a major driver behind the booming Internet of Things (IoT). Sensor-equipped devices are key to gathering data, but their limited processing power means this information must be transferred to the cloud for storage, analysis, and decision-making. While cloud computing offers convenient access to powerful resources, it can't always keep up with the real-time needs of certain IoT applications [1]. Fog computing complements rather than replaces cloud computing, collaborating to handle diverse task lengths and computations. With widespread edge-cloud adoption, user requests vary between cloud and fog nodes based on task performance metrics, requiring processing to meet user needs [2]. The different characteristics of fog and cloud, along with unpredictable user requests and resource constraints, complicated task scheduling necessitating, discussion, and attention.

Offloading and task scheduling significantly influences system performance by minimizing network overhead, maximizing resource utilization, and reducing energy consumption [3]. Task scheduling primarily involves mapping tasks to suitable resources to ensure task execution completion while meeting the QoS requirements

[4]. Despite the notable advantages of fog-cloud collaboration, task scheduling encounters challenges stemming from resource demands, task configurations, and its dynamic nature. These factors affect the QoS optimization, necessitating parameter adjustments and the selection of suitable fog and cloud resources [5]. The primary objective of optimization in task scheduling is to minimize or maximize specific functions. Optimization metrics include makespan, energy consumption, delay, and cost among others [4]. Scheduling utilizes meta-heuristic algorithms to approximate optimal solutions, often involving randomized search methods, because it is considered Nondeterministic Polynomial (NP)-complete [6]. As a result, many popular population-based algorithms are used for scheduling workflow, such as PSO, Simulated Annealing (SA), Ant Colony Optimization (ACO), and Genetic algorithm (GA) [7]. In addition, there are many more non intelligence (classical) algorithms as Round Robin (RR) [8], First Come First Serve [9] (FCFS), Max-Min [10].
Tasks are conceptualized as workflows in numerous IoT applications. A workflow comprises interdependent tasks that must be executed with designated priorities and in a specific sequence [11]. Task scheduling is depicted as Directed Acyclic Graph (DAG) in a workflow. Each node represents a task, with the weight indicating its computational cost or runtime. The prerequisite relationships between tasks are denoted by the edges of the graph [11, 12], forming the workflow. The objective may include minimizing cost, energy consumption, storage space usage, data transfer time, overall runtime, or a combination of these factors.
Building on the proposed work [13, 14], this research investigates the performance of PSO algorithm specifically for workflow scheduling within the novel three-tier IoT-cloud-fog environment illustrated in Figure 1. In contrast to [14], which focused solely on fog computing, this research investigates EPSO's effectiveness within the broader cloud-fog computing paradigm. The EPSO uses linearly decreasing inertia weight allowing particles to explore the search space greatly in the initial stages. That increases the chance of finding promising regions containing the optimal solution. As the inertia weight linearly decreases over time, the particles' exploration range contracts. This focuses their search on exploiting the identified promising region, leading to faster convergence towards the optimum. Comparative analysis of EPSO, PSO, Max-Min, and RR scheduling algorithms based on performance metrics as cost, energy consumption, and makespan demonstrates the effectiveness of this approach. The newly developed method focuses on reducing the time required to complete numerous jobs (makespan), prioritizing it as the objective function due to its significant effect on enhancing user satisfaction and increasing the productivity of computational resources. This work is implemented on FogWorkflowSim [15].

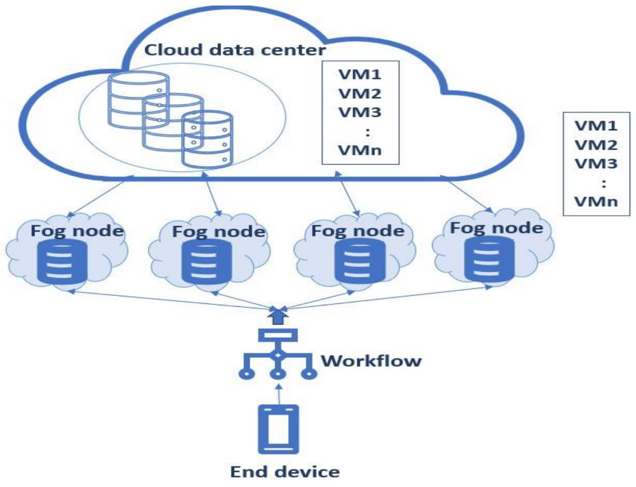

Figure 1. Paradigm of cloud-fog collaboration [4].

There are many recent researches on task offloading and scheduling in fog-cloud collaboration. Tychalas and Karatza [8] built upon the weighted round-robin algorithm to propose a dynamic probabilistic load balancing approach. This method assigns probabilities to available resources based on a combination of their computational power and how busy they currently are, as measured by key server metrics.
Subramoney and Nyirenda [13] compared cloud and fog-cloud collaboration using the well-established PSO for workflow scheduling. A weighted objective function that considers three key factors: cost, makespan, and en-

ergy consumption is presented. Additionally, they leverage the recently developed FogWorkflowSim to simulate both cloud and cloud-fog architectures.

Subramoney and Nyirenda [16] proposed a comparative assessment of population-based methods for workflow tasks scheduling in IoT-fog-cloud environments.

Pham et al. [17] declared the trade-off between makespan and cost, alongside meeting deadline constraints in task scheduling. It introduces a heuristic algorithm called Cost-Makespan aware Scheduling (CMaS) to manage these competing objectives.

M Gamal et al. [18] introduced three offloading strategies designed for IoT-fog-cloud environments, particularly targeting real-time applications. LCO is optimal for tasks requiring low latency. EBO prioritizes energy efficiency and computationally intensive tasks. EO aims to strike a balance between latency and energy consumption, optimizing resource utilization.

P. Memari et al. [19] introduced for resource allocation a tabu search method. Tabu search is chosen due to its versatility across various optimization problems and its advantages in memory and speed. The research focuses on tackling the problem of finding the most efficient allocation to optimize the utilization of resources and minimize response time. To address this challenge, a tabu search method is proposed to mitigate hardware costs. Furthermore, using meta-heuristic approaches, the study presents a latency-aware scheduling algorithm based on Virtual Machine (VM) matching. The tabu search is enhanced through the integration of Fruit Fly Optimization (FOA) algorithms and Approximate Nearest Neighbor (ANN) techniques.

Ali et al. [20] suggested a task scheduling method using the Multi-objective Optimization Problem (MOP) to minimize both makespan and cost. This approach utilizes a model incorporating Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II) to allocate tasks automatically to fog or cloud devices. The model effectively distributes the workload among cloud and fog nodes.

Fellir et al. [21] addressed task scheduling with a model of multi-agent that prioritizes tasks based on a combination of factors: priority, waiting time, and resource availability. Yadav et al. [22] addressed the cloud makespan and cost trade-off by proposing the Budget-Aware Scheduling (BAS) algorithm for sequencing applications. The technique focuses on scheduling applications on a timeline to ensure their timely execution, thereby reducing the required expenditures for utilizing cloud resources. It aims to enhance resource utilization.

Farid et al. [23] tackled the Multi-objective Optimization Problem (MOP) by employing a PSO approach based on fuzzy resource utilization in workflow scheduling. The objective is to reduce costs and makespan while ensuring reliability constraints are met. Moreover, the research simultaneously accounts for both data transportation order and task execution location. Subramoney and Nyirenda [24] introduced a method called multi-swarm particle swarm optimization (MS-PSO) to enhance the scheduling of scientific tasks in IoT-cloud-fog systems and addresses the premature convergence issue of classic PSO. Table 1 provides a concise overview of the methodologies and critical parameters addressed by each task scheduling algorithm in the mentioned references.

The paper's remaining sections are organized as follows: Section 2 contains studies on offloading and task scheduling algorithms. Section 3 details the scheduling of workflow and performance metrics. Section 4 details the concept of workflow and the proposed method. Section 5 discusses the experimental outcomes. Finally, Section 6 demonstrates the research's conclusion.

Table 1. comparison of current methods for task scheduling.

Authors	Technique used	Objective criteria
Tychalas et al. [8]	An advanced Weighted Round Robin algorithm	load balancing task execution, and time
Subramoney et al. [13]	PSO with a weighted sum objective function	Makespan, energy consumption, and cost
Subramoney et al.	Evaluation of workflow scheduling	Makespan, energy consumption, and cost

Table 1. (Continued.) comparison of current methods for task scheduling.

[16]	across cloud and fog infrastructures	
Pham et al. [17]	Collaborative task scheduling between cloud and fog	cost
Gamal, M et al. [18]	Three offloading strategies (LCO, EBO, and EO)	Makespan, energy consumption, and cost
Memari et al. [19]	latency-aware scheduling approach in cloud-fog	Latency, and cost
Ali et al. [20]	Non-dominated Sorting Genetic Algorithm II (NSGA-II)	Execution time, makespan, and energy consumption
Fellir et al. [21]	A multi-agent system	execution time, resource utilization, and energy consumption
Yadav et al. [22]	Task classification, and resource allocation	Latency, and resource utilization
Farid et al. [23]	Multi-Objective Optimization Algorithm	Cost, and execution time
Subramoney et al. [24]	Multi-Swarm PSO	Execution time, makespan, energy consumption, and load balancing

2. Methodology of workflow scheduling

2.1. The notion of Workflow

The concept of workflow application is illustrated through a DAG, represented by G = (T, E), where T represents vertices denoting tasks from t_1 to tn and edges E [16, 23] denoting task dependencies. Each edge represents data inter-task, denoted as $d_{i,j} =< t_i, t_j > \in E$, with $d_{i,j}$ denoting the size of output data from task t_i to task t_j. Task t_j begins execution after task t_i is completed. A task t_i without a parent is considered a starting task, while a task t_j without a child is regarded as an ending task. Figure 2 shows a workflow example with nine tasks. Tasks on the same level (appearing side-by-side) can be done at the same time. For instance, tasks t_2, t_3, and t_4 can all be run concurrently.

In an IoT-cloud-fog environment, workflow application offloading and scheduling involve assigning tasks within a workflow to various computing resources. These resources have unique characteristics, and the goal is to achieve optimal workflow execution by minimizing three factors: total completion time (makespan), energy consumption, and overall cost.

2.2. Performance metrics

This setup includes three primary computational resources categories of: end devices, fog servers, and cloud servers. These resource groups encompass processing and storage capacities, as well as bandwidth, memory, and the requirements of power. The computational assets within the fog and cloud segments are represented as VMs. The inclusion of end devices is justified by the fact that certain minor tasks are more efficiently handled locally, considering economic and resource efficiency, rather than being offloaded to fog and cloud servers.

This research prioritizes a single optimization goal to identify the most efficient approach for both task offloading and resource selection. This objective focuses on minimizing makespan of the workflow. However, the study also considers other factors like energy consumption and total costs.

2.2.1. Makespan

The workflow makespan, which represents the total time needed to complete the entire workflow successfully, is calculated using the following formula:

$$MS=\max\{FT_{ti},ti\in T\}-\min\{ST_{ti},ti\in T\} \quad (1)$$

, Where ST_{ti} represents the starting time and FT_{ti} signifies the task ti finishing time within a specific workflow.

2.2.2. Energy consumption

Energy consumption [25] is established using idle and active components, represented as E_{idle} and E_{active} respectively. The first one pertains to energy utilized during resource idleness, whereas the second represents energy expended during task execution. The energy expended during the idle interval [13, 25] is calculated as follows:

$$E_{idle} = \sum_{j=1}^{m} \sum_{idle_{i,k} \epsilon IDLE_{i,k}} \alpha \, f_{min\,j} \, V^2{}_{min\,j} \, L_{j,k}, \tag{2}$$

, Where $idle_{i,k}$ corresponds to a collection of periods of idle slot k on resource j, and $f_{min\,j}$ represents the frequency along with $V_{min\,j}$ that denotes the minimum voltage for resource j, correspondingly. $L_{j,k}$ represents the duration of the idle time for $idle_{i,k}$. Therefore, the active energy is determined by.

$$E_{active} = \sum_{i=1}^{n} \alpha f_i \, V^2{}_i \, (FT_{ti} - ST_{ti}), \tag{3}$$

, Where α represents a fixed number, while f_i and V_i represent the frequency and supply voltage of the task i executed resource. When the resource is in an idle state, it enters sleep mode, characterized by the lowest voltage supply and a relative frequency. The overall energy consumption (TE) across the IoT-cloud-fog system entire workflow during the execution is given by:

$$TE = E_{active} + E_{idle}. \tag{4}$$

2.2.3. Cost

This encompasses both communication and computation expenditures. Computational costs are applicable across all three types of computational resources, However, when tasks are carried out on the end device, communication costs are not incurred. The computational cost [25] associated with the utilization of resource for computing r is outlined like follows:

$$CE_i^r = pr. (FT_{ti} - ST_{ti}). \tag{5}$$

, Where the cost of unit processing is represented as pr. The communication cost, that indicates the data transmission expenditure for conveying a task's output of dimension $d_{i,j}$ from the executed resource for task i to the designated resource handling task j, is established by:

$$CC_{i,j} = trc_{i,j} \, . \, d_{i,j}, \tag{6}$$

Here, $trc_{i,j}$ represents the individual communication cost from the selected resource for task i to the allocated resource for task j. $trc_{i,j}$ is equal to 0 if both tasks are performed on the identical resource. Consequently, for a scenario involving n tasks and m computational resources, the overall cost TC can be expressed as follows:

$$TC = \sum_{i=1}^{n} \sum_{j=1}^{n} CC_{i,j} + \sum_{i=1}^{n} \sum_{j=1}^{n} CE_i^r. \tag{7}$$

This study emphasizes minimizing the time needed to complete multiple tasks (makespan) presented in equation (1), considering it as the primary objective to improve user satisfaction and enhance the productivity and efficiency of computational resources. Therefore, the objective function is defined as follow:

$$F(p) = MS \tag{8}$$

, Where p presents the assignment of a workflow's n tasks to the m available computing resources. In PSO terminology, p is known as a particle.

3. Proposed EPSO

EPSO is an extension of PSO which is a computational method proposed by Kennedy and Eberhart [26] used for solving optimization problems. Motivated by the social behavior of fish gathering or birds schooling, PSO simulates the social behavior of individuals (particles) within a swarm. Each particle denotes a potential solution to the optimization issue, and the swarm collectively searches the solution space to find the optimal solution. PSO is simple and relatively easy to understand and implement. But it suffers from premature convergence issue due to constant inertia weight. So, Linearly Decreasing Weight EPSO tackles this issue by introducing a dynamic inertia weight by using a weight that starts high and linearly decreases over time.

3.1. Particle representation

This section explains the particle denoted by "p," which is involved in the objective function in Eq. (8), before going over the rationale for the proposed EPSO. In this study, the tasks within the workflows can be arranged for implementation either at the source (end device), a cloud VM, or a fog VM. End devices do not delegate their tasks to other end devices; rather, they can only assign tasks to cloud and fog resources. Consequently, when scheduling workflows, only a single representative end device is included in the encoding process. Given that scheduling tasks in a cloud-fog environment is inherently discrete, natural numbers are employed to encode individuals for the proposed EPSO algorithm.
This section focuses on the concept of particles. These particles represent mappings between tasks and the resources they'll be executed on. Each particle, "p", is a one-dimensional vector with a length equal to the total number of tasks (T) in the workflow. Each element (position) within this vector indicates the specific VM assigned to a particular task. The value at each position is an integer ranging from 1 to R, where R represents the total number of available resources in the FogWorkflowSim environment.
For example, imagine Figure 3 depicts a workflow with T=8 tasks that need to be mapped to an IoT-cloud-fog environment with R=5 available resources (including 2 cloud VMs and 2 fog VMs). In this scenario, the particle p = {3, 5, 2, 4, 1, 3, 2, 5} represents a potential mapping solution. Each value in the particle vector corresponds to a task, and the value itself indicates the assigned VM (between 1 and 5) for that specific task.

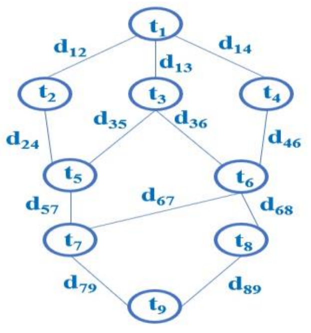

Figure 2. Workflow of nine tasks example [4].

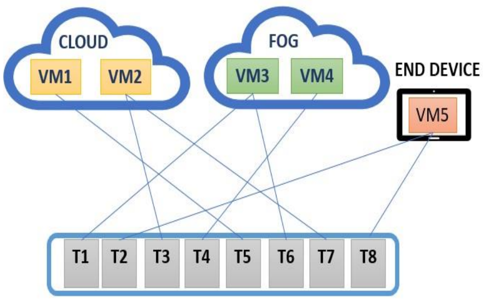

Figure 3. Workflow scheduling example [4].

3.2. Workflow Scheduling Using EPSO

EPSO is a technique for optimizing how tasks within a scientific workflow are assigned to computing resources. The goal is to find the most efficient allocation that considers reducing in makespan as objective function. In this approach, particles represent a population of individuals; they navigate through a given solution space based on their current positions x^k_i and velocities v^k_i for the kth iteration. The quality of each particle is assessed by applying a predetermined fitness function that is specific to the optimization issue. Each particle's best-known personal position $pBest_i$ affects how it moves as well as the best-known global position gBest for the whole swarm.

This iterative process guides the swarm towards the optimal position. EPSO's overview is explained in Algorithm 1 as:

- Initialize EPSO parameters like the number of particles (N), search space boundaries, minimal and maximal inertia weight, learning factors, and maximum iterations (G).
- Generate a swarm of particles with random positions (representing task schedules) and velocities within the search space.
- Calculate the fitness of each particle's schedule using makspan metric using Eq. (8).
- For each particle, update its personal best position (pbest) if the current schedule has better fitness.
- Find which particle has the best fitness (gbest) in the entire population.
- For each iteration do:
    - ➢ Linearly decrease the inertia weight (ω) from its maximal value (ω_{begin}) to a minimal (ω_{end}) as:

$$\omega^t = \omega_{end} + (\omega_{begin} - \omega_{end}) \times \frac{t_{max-t}}{t_{max}} \tag{9}$$

, Where ω_{begin} is a starting value of inertia weight, ω_{end} is finishing value, t_{max} is the total iteration number and t represents the current iteration.

- ➢ Update the velocity of each particle based on its current velocity, the difference between its position and pbest/gbest, and the new inertia weight as follow:

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 (pBest_i - X_i^k) + c_2 r_2 (gBest_i - X_i^k) \tag{10}$$

, where i represents Particle number, ω is the inertia weight; r_1 and r_2 are represented randomly between (0,1); and c_1 and c_2 are the learning factors.

- ➢ Update the position of each particle based on its current position and velocity.as follow:

$$x_i^{k+1} = x_i^k + v_i^k \tag{11}$$

- ➢ Calculate the fitness of the particles' schedules after position update.
- ➢ If a particle's new position has better fitness, update its pbest.
- ➢ Verify if the total number of iterations has been reached.
- ➢ If the termination condition is met, exit the loop and return the best solution (the schedule).

Due to the simplicity of the Socio-Cognitively Inspired standard PSO [13], this work adopts it, but on other hand it suffers from Premature Convergence. Furthermore, in this technique, to schedule particle positions, first discretize each dimension variable into an integer in the fog server set p= {1, 2, …., n} as in previous subsection. This ensures that particle positions match task scheduling. Secondly, decreasing ω linearly over iterations by using a weight that starts high and linearly decreases over time. This can help in fine-tuning the search behavior of particles during different stages of the optimization process. It ensures that particles continue to explore the search space, preventing premature convergence.

EPSO achieves a better balance between exploration and exploitation compared to standard PSO with a fixed weight. This can lead to finding better schedules that optimize resource utilization and minimize execution time. On other hand a very rapid decrease in ω might limit exploration too early, hindering the discovery of diverse solutions.

Algorithm 1 depicts the EPSO optimization method. The parameters N represents the number of particles and G denotes the number of generations. The remaining parameters are defined in previous subsections. The algorithm begins with the initialization of N, c_1, c_2, ω, and G. The process involves constructing N particles and evaluating them using the Fogworkflowsim toolbox [15]. To determine the global and personal optimal values, the value of fitness function is evaluated. Then the algorithms start a repeated process for G generations. Based on improved values, the system modifies the personal and global best values.

Algorithm 1 EPSO algorithm

```
1:  Set   G, $c_1$, $c_2$, $\omega$begin, $\omega$end, and N      ► Inputs
2:  gBest and its evaluation F(gBest)          ► outputs
3:  N particles randomly generated
4:     F(gBest) = 0
5:  for i ≤ N do
6:  Call up the workflow scheduler
7:  Fitness value computed, F($x_i$) for particle i using Eq.8
8:  set $pBest_i$ equal $x_i$
9:  F($pBest_i$) = F($x_i$)
10: if F($x_i$) > F(gBest) then
11: gBest = $x_i$
12: F(gBest) = F($x_i$)
13: end if
14: end for
15: t = 0                                              ► initialize t
16: while t ≤ G do
17: for k ≤ N do
18: Update $\omega t$ as Eq. 9
19:   Update $v^k$ and $x^k$ by applying Eq. 10 and Eq. 11
20: Call up the workflow scheduler
21: Fitness value computed, F($x^k$), as in Eq. 8
22: if F($x^k$) > F($pBest^k$) then
23: $pBest^k$ = $x^k$
24: F($pBest^k$) = F($x^k$)
25: end if
26: if F($x^k$) > F(gBest) then
27: gBest =$x^k$
28: F(gBest) = F($x^k$)
29: end if
30: end for
31: t=t+1
32: end while
```

4. Performance evaluation

This section commences by providing an overview of the workflow models and outlines the setup of the simulation environment using the FogWorkflowSim Toolkit [15]. Following these initial details, the section proceeds to present the experimental results and engage in a discussion thereof.

4.1. Workflow models

This research employs five established scientific workflows [27] from diverse scientific fields to thoroughly assess the effectiveness of the proposed method. These scientific workflows, described by the Pegasus framework [28], utilize XML files containing DAG representations as input for the simulations. Workflows involve several tasks, dependencies, run-times, and data transfers. Fig. 4 illustrates the workflows in a graphical arrangement.

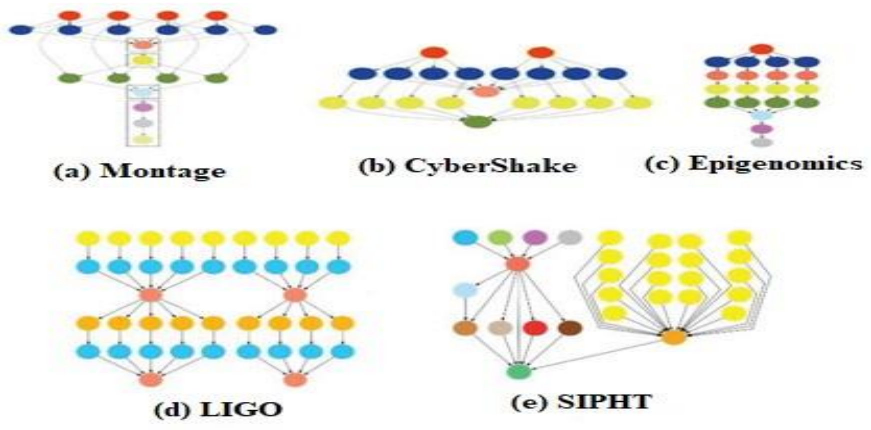

Figure 4. Scientific workflows' structure [4].

The workflows are outlined below:

1) **Montage Workflow**: This workflow simulates an application of astronomy that generates custom mosaics of the sky from utilizing several input images.
2) **CyberShake Workflow**: This workflow is used to assess and characterize earthquake hazards posing a threat to a specific region.
3) **Epigenomics Workflow**: Applied in the field of bioinformatics, this workflow automates various operations involved in processing genome sequences.
4) **LIGO Workflow**: It's a detector that uses lasers to measure incredibly tiny vibrations in spacetime, with the goal of directly observing gravitational waves.
5) **Sipht Workflow**: Developed to automate the search for small RNA (sRNA) encoding genes across all bacterial replications.

4.2. Simulation environment

The FogWorkflowSim simulator is executed using the Eclipse Java IDE. The simulations are conducted on a computer system equipped with a 64-bit Windows 10 operating system, an Intel (R) Core (TM) i7- M 640 @ 2.80 GHz, and 6 GB of RAM. Each algorithm was run with 10 particles.

In EPSO, learning factors C_1 and C_2 are set to 2, while the initial and final inertia weights (ωbegin and ωend) are 0.8 and 0.3, respectively. To evaluate average performance, simulations are carried out 10 times for each workflow. PSO uses an inertia weight of 1 and learning factors of 2 for C_1 and C_2. The simulation environment involved ten cloud VMs, six fog VMs, and one end device. The specific characteristics of each server in the three-layer IoT-fog-cloud architecture, along with the detailed parameter configurations, are presented in Table 2.

Table 2. IoT-cloud-fog environment parameter setting.

Parameters	End device	Fog VM	Cloud VM
Rate of Processing (MIPS)	1000	1300	1600
Task execution Cost ($)	0	0.48	0.96
Communication Cost ($)	0	0.01	0.02
Busy Power (MW)	700	800	1600
Idle Power (MW)	30	40	1300
Uplink Bandwidth (Mbps)	20	10	1
Downlink Bandwidth (Mbps)	40	10	10

4.3. Simulation results

The performance of EPSO algorithm is evaluated against three other scheduling methods: PSO, Max-Min, and Round- RR. All four algorithms are compared under the same simulated conditions to ensure a fair assessment. Figures 5, 6, and 7 show how different scheduling algorithms perform on Montage, LIGO, and SIPHT work-flows when the number of tasks varies (100, 300, and 500 tasks). Interestingly, all algorithms seem to perform about the same for Montage workflows regardless of the number of tasks.
EPSO algorithm demonstrably outperformed other compared methods (PSO, Max-Min, and RR) on all three workflow benchmarks: Montage, LIGO, and SIPHT. Specifically, EPSO achieved significant reductions in makespan. On the Montage workflow, EPSO's average improvement was 36.78% over PSO, 44.05% over Max-Min, and a substantial 44.66% over RR. In LIGO execution, EPSO impressively reduced the makespan by 36.57% compared to PSO, 24.39% compared to Max-Min, and 27.16% compared to RR. Finally, for the SIPHT workflow, EPSO continued to exhibit superior performance with improvements in makespan of 4.78% com-pared to PSO, 20.23% compared to Max-Min, and a noteworthy 32.24% improvement over RR.
For different workflows, EPSO ranks the best performance once 300 tasks are completed. For Montage, LIGO, and SIPHT, it reduces makespan compared to PSO by 8.47%, 28.29%, and 35.91%, respectively. It decreases makespan by 23.92%, 37.46%, and 25.38% in comparison to Max-Min. Ultimately, it achieves reductions in makespan of 24%, 27.06%, and 32.97% for different workflows when compared to RR.
Finally, when executing 500 tasks, EPSO displays the most favorable performance in terms of makespan be-cause it decreases makespan more than other algorithms for all workflows. It reduces makespan than PSO by 12.92%, 21.23%, and 40.10% for Montage, LIGO, and SIPHT, respectively; Max-Min by 23.05%, 23.38%, and 36.70%; and RR by 23.55%, 8.19%, and 43.45%.
Figures 8, 9, and 10 show the energy consumption metric's average performance results for the three workflows across varying task sizes: 100, 300, and 500 tasks. For 100 task execution Montage exhibits the smallest energy consumption. Proposed EPSO outperforms other algorithms except for SIPHT workflow PSO shows the best performance. It has more favorable performance by decreasing energy consumption than PSO by 38.50% and 61.02% for Montage and LIGO, while PSO excels EPSO for SIPHT by decreasing energy consumption 7.03% than it. EPSO presents better performance by 58.84% and 44.18% than Max-Min, and RR respectively for LIGO workflow. Finally, Max-Min presents the best performance for SIPHT. It decreases energy consumption com-pared to
EPSO by 16.54%. While RR presents the lowest performance it shows increasing in energy consumption com-pared to EPSO by 7.83%.
For the execution of 300 tasks, the Max-Min algorithm performs the poorest across three workflows, while EPSO consistently surpasses other algorithms. In the Montage workflow, EPSO reduces energy consumption by 10.93%, 27.06%, and 19.28% compared to PSO, Max-Min, and RR, respectively. EPSO also achieves the best performance in the LIGO workflow, lowering energy consumption by 18.96%, 33.35%, and 22.97% compared to PSO, Max-Min, and RR, respectively. Similarly, in the SIPHT workflow, EPSO reduces energy consumption by 34.97%, 54.29%, and 40.77% compared to PSO, Max-Min, and RR. These results indicate that EPSO is the most favorable algorithm for reducing energy consumption in the 300-task case study for both the LIGO and SIPHT workflows.
With the execution of 500 tasks, the Max-Min algorithm consistently demonstrates the weakest performance compared to other algorithms. Meanwhile, PSO for the SIPHT workflow yields the lowest efficiency. The mod-ified EPSO algorithm outperforms the other algorithms across various workflows, with the exception of the LIGO workflow. Specifically, in the Montage workflow, EPSO achieves significant energy savings, reducing consumption by 63.62%, 70.77%, and 20.19% compared to PSO, Max-Min, and RR, respectively. In the LIGO workflow, EPSO reduces energy consumption by 20.22% and 21.07% compared to PSO and Max-Min, though RR achieves a further reduction of 9.19% compared to EPSO.

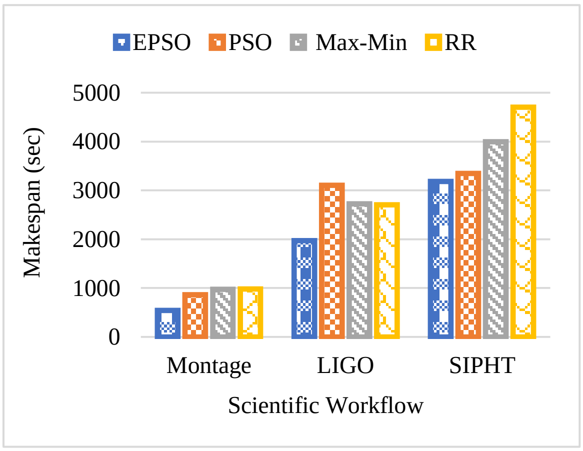

Figure 5. For 100 tasks, makespan comparison.

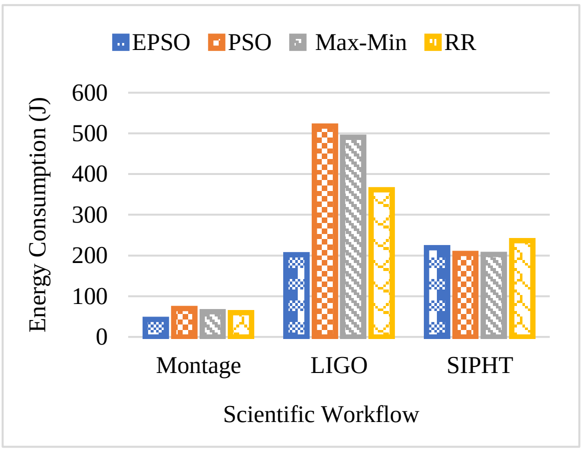

Figure 8. For 100 tasks, energy consumption comparison.

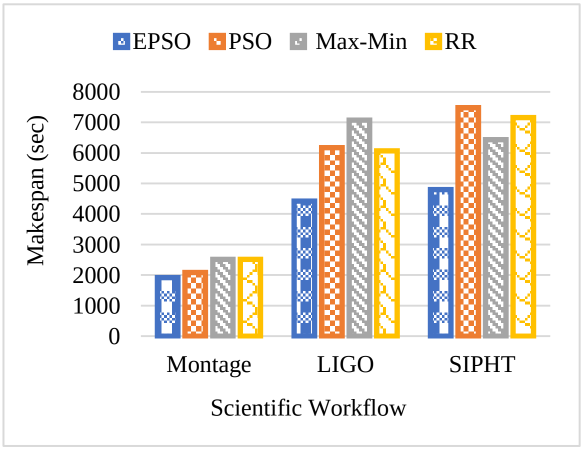

Figure 6. For 300 tasks, makespan comparison.

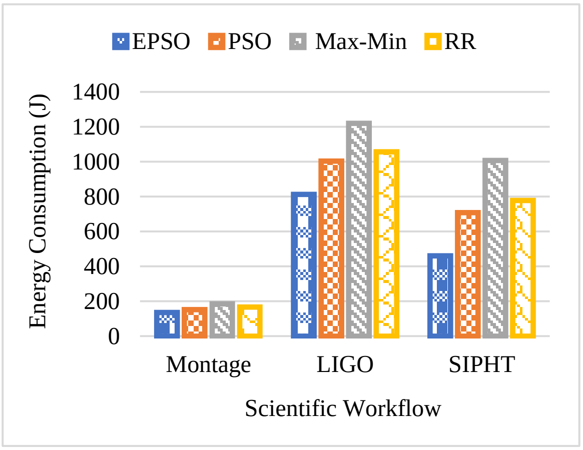

Figure 9. For 300 tasks, energy consumption comparison.

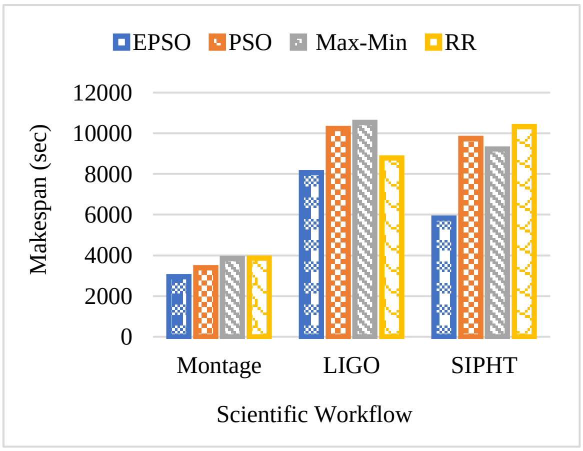

Figure 7. For 500 tasks, makespan comparison.

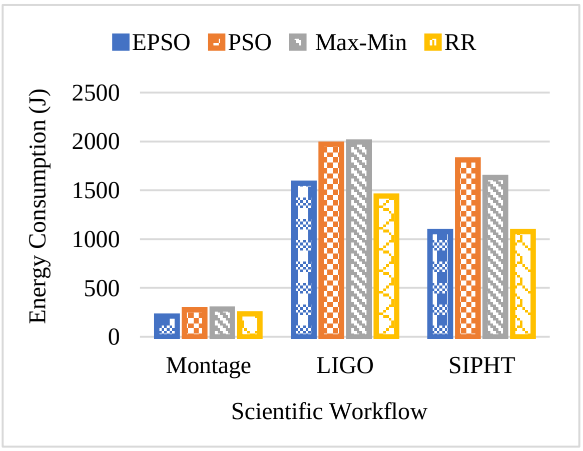

Figure 10. For 500 tasks, energy consumption comparison.

Figures 11, 12, and 13 illustrate the average performance results of the total cost metric for the five workflows across varying task sizes: 100, 300, and 500 tasks. As the number of tasks increases, Montage consistently presents the lowest cost. For the LIGO and SIPHT workflows, there is a slight increase in cost when the quantity of tasks increases.

For the execution of 100 tasks, the proposed EPSO outperforms PSO in reducing costs by 18.22%, 10.88%, and 3.91% for Montage, LIGO, and SIPHT, respectively. However, PSO demonstrates better performance in cost reduction compared to EPSO for LIGO by 3.51%. EPSO achieves higher performance than Max-Min by reducing the total cost by 39.83%, 6.50%, and 24.90% for Montage, LIGO, and SIPHT respectively. It also achieves better cost reduction than RR by 45.74%, 2.71%, and 19.42% for Montage, LIGO, and SIPHT, respectively.

For 300 tasks execution, when it comes to cost reduction, the suggested EPSO performs better than the other three algorithms. It achieves higher reductions compared to PSO by 0.39%, 0.81%, and 2.10%, compared to Max-Min by 45.91%, 16.06%, and 12.23%, and compared to RR by 46.85%, 12.72%, and 8.61% for Montage, LIGO, and SIPHT, respectively. Finally, for 500 tasks execution, the proposed EPSO continues to outperform the other three algorithms in terms of reducing the total cost. It achieves higher reductions compared to PSO by 6.94%, -0.61% (indicating that in the case of executing the LIGO workflow, PSO outperforms EPSO in terms of cost reduction), and 5.11% for Montage, LIGO, and SIPHT respectively. EPSO reduces the total cost compared to Max-Min by 49.22%, 22.13%, and 15.85%, and compared to RR by 50.95%, 19.26%, and 12.96% for Montage, LIGO, and SIPHT, respectively. Finally, when executing 500 tasks, the proposed EPSO algorithm outperforms the other three algorithms in terms of total cost reduction. Compared to PSO, EPSO achieves a cost reduction of 6.94% for Montage, is 0.61% less efficient for LIGO (indicating PSO outperforms EPSO in this case), and is 5.11% better for SIPHT. When compared to Max-Min, EPSO shows improvements of 49.22%, 22.13%, and 15.85% for Montage, LIGO, and SIPHT, respectively. Similarly, compared to RR, EPSO yields cost reductions of 50.95%, 19.26%, and 12.96% for the three workflows.

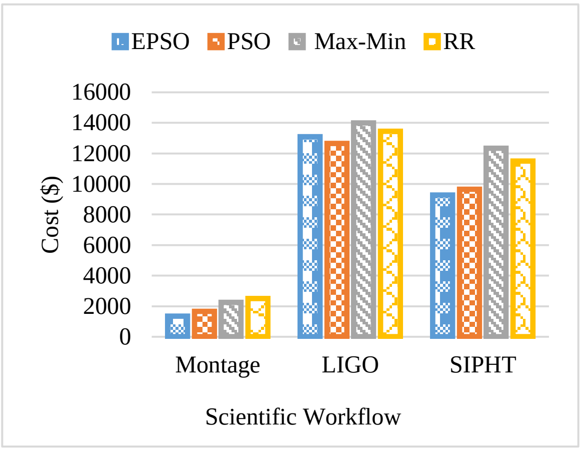

Figure 11. For 100 tasks, cost comparison.

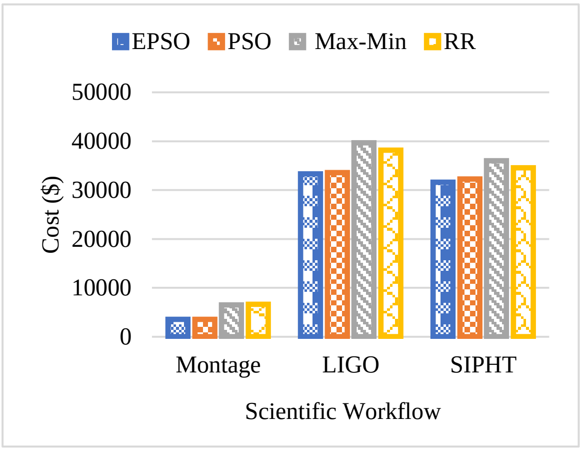

Figure 12. For 300 tasks, cost comparison.

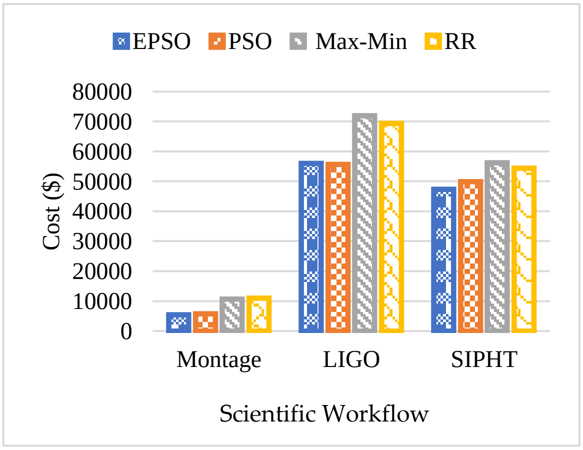

Figure 13. For 500 tasks, cost comparison

The previous results clearly demonstrate the superiority of the proposed EPSO algorithm over other algorithms across a variety of workflows, particularly in optimizing the makespan metric, which is the focus of this research. EPSO not only outperforms classical PSO but also addresses its premature convergence issue effectively. Moreover, it's noticeable that EPSO rarely shows the worst performance among the evaluated algorithms. These outcomes are influenced by the unique characteristics of each workflow instance, particularly as the number of

tasks increases, leading to higher metric values due to longer runtimes and larger data sizes in the workflow DAGs.
Workflows in CyberShake and Epigenomics are notable for involving extensive and time-intensive tasks, resulting in substantially higher performance metrics. These workflows, along with a case study and the performance of the algorithms, are detailed in Table 3. For the execution of 100 tasks, the proposed EPSO outperforms PSO, Max-Min, and RR in reducing Makespan by 0.57%, 0.57%, and 0.57%, respectively for Cybershake and by 34.28%, 53.23%, and 41.22%, respectively for Epigenomics. In terms of energy consumption, EPSO surpasses PSO, Max-Min, and RR by 71.50%, 0.19%, and 0.93%, respectively, for Cybershake, and by 32.40%, 69.09%, and 62.51%, respectively, for Epigenomics. Regarding cost, EPSO reduces it by 10.88%, 41.99%, and 41.88%, respectively, for Cybershake, but for Epigenomics, PSO outperforms EPSO by 0.58%, while EPSO still outperforms Max-Min and RR, reducing cost by 21.38% and 15.80%, respectively.
For the execution of 300 tasks, the proposed EPSO outperforms PSO, Max-Min, and RR in reducing Makespan by 0.63%, 8.45%, and 8.73%, respectively, for Cybershake, and by 36.26%, 31.57%, and 20.70%, respectively, for Epigenomics. In terms of energy consumption, although PSO surpasses EPSO by 0.41% for Cybershake, EPSO outperforms Max-Min and RR by 7.20% and 6.14%, respectively. For Epigenomics, EPSO excels in reducing energy consumption, outperforming PSO, Max-Min, and RR by 43.49%, 39.91%, and 8.10%, respectively. Regarding cost, EPSO reduces it by 31.20%, 47.24%, and 49.33%, respectively, for Cybershake, and by 3.07%, 20.05%, and 21.02%, respectively, for Epigenomics compared to PSO, Max-Min, and RR.
Finally, for the execution of 500 tasks, the proposed EPSO outperforms PSO, Max-Min, and RR in reducing Makespan by 1.76%, 10.71%, and 10.86%, respectively, for Cybershake, and by 29%, 37.06%, and 29.28%, respectively, for Epigenomics. In terms of energy consumption, EPSO surpasses PSO, Max-Min, and RR by 0.19%, 10.05%, and 7.60%, respectively, for Cybershake. For Epigenomics, EPSO excels in reducing energy consumption, outperforming PSO, Max-Min, and RR by 33.36%, 33.46%, and 20.38%, respectively. Regarding cost, EPSO reduces it by 36.94%, 58.39%, and 60.71%, respectively, for Cybershake, and by 1.49%, 28.45%, and 24.25%, respectively, for Epigenomics compared to PSO, Max-Min, and RR.

Table 3. Algorithms performance for cybershake and epigenomics workflows.

Performance	**Workflow**	**Cybershake**				**Epigenomics**			
Metrics	Algorithm	EPSO	PSO	Max-Min	RR	EPSO	PSO	Max-Min	RR
	100 Tasks	**103314**	103904	103903	103902	**44417**	67583	94963	75567
Makespan (sec)	300 Tasks	**99668**	100303	108862	109201	**23734**	37235	34682	29929
	500 Tasks	**99731**	101522	111699	111886	**113677**	160111	180609	160736
	100 Tasks	**5211**	18287	5221	5260	**4701**	6954	15208	12540
Energy consumption (J)	300 Tasks	5185	**5164**	5587	5524	**3699**	6595	6156	4025
	500 Tasks	**5280**	5290	5870	5714	**17251**	25886	25924	21668
	100 Tasks	**455138**	510703	784527	783124	270056	**268479**	343512	320723
Cost ($)	300 Tasks	**143190**	208129	271404	282608	**155964**	160898	195078	197453
	500 Tasks	**121352**	192453	291630	308851	**921779**	935722	1288316	1216840

Table 3 demonstrates that the EPSO algorithm consistently outperforms other algorithms in reducing the total completion time (makespan) for both workflows, even as the tasks increase. Amongst the algorithms tested on the Cybershake workflow, EPSO consistently outperformed the others, particularly in terms of metrics related to increasing task numbers. While all algorithms achieved similar results for makespan (total completion time), EPSO demonstrably reduced energy consumption for scenarios with 100, 300, and 500 tasks. Furthermore, EPSO emerged as the most cost-effective option as the number of tasks increased.EPSO remains impressive for Epigenomics workflows. It consistently reduces completion time (makespan) compared to other algorithms for all task sizes (100, 300, and 500). EPSO also dominates in saving energy, especially as tasks increase. There's one exception: for total cost with only 100 tasks, PSO seems to perform slightly better as shown in Table 3.

5. Conclusions

This study proposes a novel approach called Modified Particle Swarm Optimization (EPSO), specifically designed for managing tasks and scheduling scientific workflows within a complex environment that combines IoT devices, cloud computing, and fog computing. The core reason for developing EPSO lies in overcoming a frequent issue with the traditional PSO technique: its tendency to converge on solutions too quickly. PSO's simplicity and ease of use have made it a popular choice for scientific workflow scheduling; however, this study emphasizes the importance of handling delay-sensitive applications where timing and latency are crucial aspects for proper function. The innovation of EPSO lies in its ability to adjust the inertia weight throughout the process. This allows EPSO to dynamically adapt its behavior, striking a better balance between exploring a wider range of solutions (exploration) and focusing on promising areas (exploitation). This dynamic approach has the potential to improve both the speed at which EPSO converges on a solution and the overall quality of the solution itself. EPSO surpasses all other methods in minimizing makespan across all tested workflows. Additionally, EPSO demonstrates competitive performance in other metrics such as energy consumption and cost.

In the future, our research aims to further refine EPSO and explore the application of a wider range of algorithms for tackling task offloading and scheduling challenges. We plan to utilize a weighted sum objective function, which will allow for a more nuanced analysis of trade-offs between different factors. Additionally, to enhance the practicality of EPSO in real-world scenarios, we intend to incorporate constraints such as budget limitations, deadlines, and resource limitations.

References

1. Hoseiny, F., Azizi, S., & Dabiri, S. (2020, September). Using the power of two choices for real-time task scheduling in fog-cloud computing. In 2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT) (pp. 18-23). IEEE. doi: 10.1109/SCIOT50840.2020.9250197.
2. Tyagi, R., & Gupta, S. K. (2018). A survey on scheduling algorithms for parallel and distributed systems. In Silicon Photonics & High Performance Computing: Proceedings of CSI 2015 (pp. 51-64). Springer Singapore.
3. Chen, L., Guo, K., Fan, G., Wang, C., & Song, S. (2020). Resource constrained profit optimization method for task scheduling in edge cloud. IEEE Access, 8, 118638-118652, doi: 10.1109/ACCESS.2020.3020225.
4. Varshney, P., & Simmhan, Y. (2020). Characterizing application scheduling on edge, fog, and cloud computing resources. Software: Practice and Experience, 50(5), 558-595.
5. Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K. K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. IEEE Transactions on Cloud Computing, 10(4), 2294-2308.
6. Akbari, M., Rashidi, H., & Alizadeh, S. H. (2017). An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. Engineering Applications of Artificial Intelligence, 61, 35-46.
7. Saif, F. A., Latip, R., Derahman, M. N., & Alwan, A. A. (2022, October). Hybrid meta-heuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment. In Proceedings of the future technologies conference (pp. 16-43). Cham: Springer International Publishing.
8. Tychalas, D., & Karatza, H. (2020, November). An advanced weighted round robin scheduling algorithm. In Proceedings of the 24th Pan-Hellenic Conference on Informatics (pp. 188-191). https://doi.org/10.1145/3437120.3437304.
9. Azizi, S., Shojafar, M., Abawajy, J., & Buyya, R. (2022). Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. Journal of network and computer applications, 201, 103333.

10. Murad, S. S., Badeel, R. O. Z. I. N., Salih, N. A. S. H. A. T., Alsandi, A., Faraj, R., Ahmed, A. R., ... & Alsandi, N. (2022). Optimized Min-Min task scheduling algorithm for scientific workflows in a cloud environment. J. Theor. Appl. Inf. Technol, 100(2), 480-506.
11. Versluis, L., & Iosup, A. (2021). A survey of domains in workflow scheduling in computing infrastructures: Community and keyword analysis, emerging trends, and taxonomies. Future generation computer systems, 123, 156-177.
12. Chen, G., Qi, J., Sun, Y., Hu, X., Dong, Z., & Sun, Y. (2023). A collaborative scheduling method for cloud computing heterogeneous workflows based on deep reinforcement learning. Future Generation Computer Systems, 141, 284-297.
13. Subramoney, D., & Nyirenda, C. (2021, November). Pso-based workflow scheduling: A comparative evaluation of cloud and cloud-fog environments. In South-ern Africa Telecommunication Networks and Applications Conference (SATNAC) (Vol. 2021, pp. 258-262).
14. He, J., & Bai, W. (2023, February). Computation offloading and task scheduling based on improved integer particle swarm optimization in fog computing. In 2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE) (pp. 633-638). IEEE.
15. Liu, X., Fan, L., Xu, J., Li, X., Gong, L., Grundy, J., & Yang, Y. (2019, November). FogWorkflowSim: An automated simulation toolkit for workflow performance evaluation in fog computing. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE) (pp. 1114-1117). IEEE.
16. Subramoney, D., & Nyirenda, C. N. (2020, December). A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments. In 2020 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 760-767). IEEE.
17. Pham, X. Q., Man, N. D., Tri, N. D. T., Thai, N. Q., & Huh, E. N. (2017). A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. International Journal of Distributed Sensor Networks, 13(11), 1550147717742073.
18. Gamal, M., Awad, S., Abdel-Kader, R. F., & Abd El Salam, K. (2024). Efficient offloading and task scheduling in internet of things-cloud-fog environment. *International Journal of Electrical and Computer Engineering (IJECE)*, *14*(4), 4445-4455. http://doi.org/10.11591/ijece.v14i4.pp4445-4455.
19. Memari, P., Mohammadi, S. S., Jolai, F., & Tavakkoli-Moghaddam, R. (2022). A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture. The Journal of Supercomputing, 78(1), 93-122. https://doi.org/10.1007/s11227-021-03868-4.
20. Ali, I. M., Sallam, K. M., Moustafa, N., Chakraborty, R., Ryan, M., & Choo, K. K. R. (2020). An automated task scheduling model using non-dominated sorting genetic algorithm II for fog-cloud systems. IEEE Transactions on Cloud Computing, 10(4), 2294-2308.
21. Fellir, F., El Attar, A., Nafil, K., & Chung, L. (2020, February). A multi-Agent based model for task scheduling in cloud-fog computing platform. In 2020 IEEE international conference on informatics, IoT, and enabling technologies (ICIoT) (pp. 377-382). IEEE.
22. Yadav, A. M., Sharma, S. C., & Tripathi, K. N. (2021). A Two-Step Technique for Effective Scheduling in Cloud–Fog Computing Paradigm. In Advances in Computational Intelligence and Communication Technology: Proceedings of CICT 2019 (pp. 367-379). Springer Singapore. https://doi.org/10.1007/978-981-15-1275-9_30
23. Farid, M., Latip, R., Hussin, M., & Hamid, N. A. W. A. (2020). Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. IEEE Access, 8, 24309-24322.
24. Subramoney, D., & Nyirenda, C. N. (2022). Multi-swarm PSO algorithm for static workflow scheduling in cloud-fog en-

vironments. IEEE Access, 10, 117199-117214.

25. Yassa, S., Chelouah, R., Kadima, H., & Granado, B. (2013). Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. The Scientific World Journal, 2013(1), 350934.
26. Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol. 4, pp. 1942-1948). ieee.
27. Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M. H., & Vahi, K. (2008, November). Characterization of scientific workflows. In 2008 third workshop on workflows in support of large-scale science (pp. 1-10). IEEE.
28. The Pegasus Website. Accessed: Jan. 10, 2022. [Online]. Available: https://pegasus.isi.edu/