Suez Canal Engineering Energy and Environmental Science Journal
Faculty of Engineering – Suez Canal University

Year 2024, Volume 2, NO. 2, pages 17– 26

Multi-objective Improved Particle Swarm Optimization for Efficient Offloading Algorithm in Fog-Cloud Collaboration

Samar Awad[1], Marwa Gamal[1], Khaled Abd El Salam[1], and Rehab F. Abdel-Kader[2]
1 Electrical Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt, email: Samar_Awad@eng.suez.edu.eg
1 Electrical Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt, email: marwa_gamal@eng.suez.edu.eg
1 Electrical Department, Faculty of Engineering, Suez Canal University, Ismailia, Egypt, email: Khaled.abdelsalam@eng.suez.edu.eg
2 Electrical Department, Faculty of Engineering, Port Said University, Port Said, Egypt, email: rehabfarouk@eng.psu.edu.eg
DOI: 10.21608/sceee.2024.305699.1034

Article Info:

Article History:
Received: 19\07\2024
Accepted: 29\08\2024
Published: 30\10\2024

DOI:
10.21608/sceee.2024.305699.1034,

Abstract

The surge in Internet of Things (IoT) devices and their diverse applications generates massive data volumes requiring substantial processing power, demanding efficient scientific workflow execution across resource-constrained edge devices, fog nodes, and the cloud. Efficiently matching workflow tasks with resources is crucial for minimizing total completion time (makespan), energy consumption, and cost, particularly in delay-sensitive applications. However, achieving this optimal allocation remains a challenge. To tackle this challenge, we introduce a novel multi-objective Improved Particle Swarm Optimization (IPSO) algorithm. We evaluate the performance of the IPSO algorithm against standard PSO. IPSO's effectiveness is assessed through simulations employing the Montage scientific workflow and a varying number of tasks, scaling up to 500. Simulations demonstrate that IPSO outperforms PSO in minimizing completion time (makespan), energy consumption, and total cost. This advantage becomes more pronounced as the number of tasks in the workflow increases, suggesting IPSO's efficacy in handling larger and more complex scientific workflows. For instance, with 500 tasks, IPSO demonstrably reduced makespan, energy consumption, and total cost compared to PSO by 15.11%, 24.02%, and 1.42%, respectively.

Keywords: *Cloud Computing, Fog Computing, Scientific Workflow, Task Scheduling*

Suez Canal Engineering, Energy and Environmental Science Journal (2024), ***2***(2), 17-26.

1. Introduction

The Internet of Things (IoT) era has caused in an increase of resource-constrained devices generating real-time data. Cloud computing, while adept at handling large-scale data processing, struggles with the latency and bandwidth demands of these applications [1]. Fog computing bridges this gap by providing a distributed computing layer at the network's edge. This facilitates real-time processing closer to data sources, fostering efficient management of delay-sensitive tasks in the IoT tier. User demands differ between cloud and fog nodes depending on job performance metrics as edge-cloud adoption increases, necessitating processing to satisfy user needs [2]. It's important to talk to the differences between fog and cloud computing, as well as erratic user requests, resource constraints, and challenging task offloading and scheduling.

System performance is greatly affected by offloading tasks that are accomplished by minimizing network overhead, maximizing reducing power consumption, and resource utilization [3]. Optimizing resource allocation, offloading, and task scheduling within this combined IoT-cloud-fog environment presents a significant challenge. These involve assigning tasks to the most suitable resources, ensuring they are completed while meeting quality of service (QoS) requirements [4]. However, the dynamic nature of the environment, along with varying task configurations and resource demands, pose challenges to task scheduling [5]. These factors make it difficult to optimize QoS, requiring adjustments and careful selection of cloud and fog resources. The primary goal of optimization in task scheduling is to find the best possible solution that can be suitable for performance metrics like delay, energy consumption, total completion time (makespan), and cost. Nondeterministic polynomial (NP) completion is the standard for scheduling. To approximate optimal solutions, it therefore makes use of meta-heuristic algorithms, which frequently involve randomized search techniques [6]. Optimizing resource allocation and task scheduling within this combined IoT-cloud-fog environment presents a significant challenge. Many scheduling workflow population-based algorithms emerge as a compelling technique for tackling this issue as PSO, Genetic algorithm (GA), Ant Colony Optimization (ACO), and Simulated Annealing (SA) [7].

In many IoT applications, tasks are organized as workflows. A workflow consists of interdependent tasks that need to be carried out in a particular order and with specific priorities [8]. In this context, task scheduling is represented as a directed acyclic graph (DAG), where each node symbolizes a task and its weight reflects the task's runtime or computational cost. The edges of the graph illustrate the prerequisite relationships between tasks [8, 9], thereby shaping the workflow. The goal may involve minimizing cost, energy consumption, storage space usage, data transfer time, overall runtime, or a combination of these factors.

According to a recent comprehensive study on QoS requirements in workflow scheduling approaches, PSO based optimization [10], [11], [12] is the most widely used strategy due to its simplicity. However, the PSO algorithm can become trapped in local optima, hindering its effectiveness. This paper proposes a modification to address this issue, creating multi-objective IPSO specifically suited for three-tier IoT-cloud-fog environment illustrated in Fig. 1. The main idea of this work is inspired by the method presented in [11] but it extended it to fog-cloud computing instead fog computing only. The IPSO employs a linearly decreasing inertia weight, which enables particles to extensively explore the search space during the initial stages. This broad exploration increases the likelihood of discovering promising regions that may contain the optimal solution. As time progresses and the inertia weight decreases linearly, the range of exploration for the particles gradually contracts. This narrowing of the search range focuses the particles on exploiting the identified promising region, leading to faster convergence towards the optimal solution. Multi-objective IPSO is designed to tackle problems with multiple, conflicting objectives. It aims to identify a set of solutions that represent the best trade-offs between these objectives.

The efficacy of this approach is demonstrated by a comparative study of multi-objective IPSO and PSO scheduling algorithms with an increase in the number of tasks and based on performance parameters including makespan, energy consumption, and cost. Because the newly created method has a considerable impact on improving user satisfaction and computing resource productivity, it emphasizes minimizing the summation of makespan, energy consumption, and total cost as the objective function. FogWorkflowSim is used to implement this work [13]. The paper's remaining sections are organized as follows: Section 2 reviews studies on offloading and task scheduling algorithms. Section 3 elaborates on the scheduling of workflows and performance metrics. Section 4 explains the concept of workflow and the proposed method. Section 5 discusses the experimental outcomes. Finally, Section 6 presents the research’s conclusion.

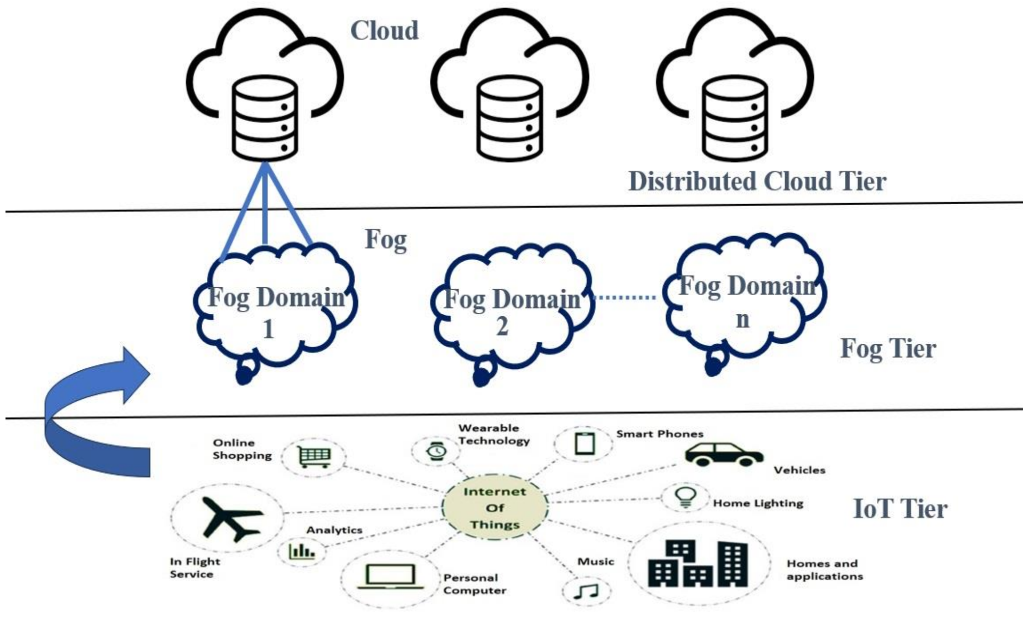


Fig. 1 IoT- Fog-Cloud Structure.

2. Related work

This review examines recent research on offloading and scheduling tasks within cloud-fog computing environments. It emphasizes the key algorithms and approaches used for this function. In response to the difficulties of large-scale task scheduling, a new real-time randomized algorithm is presented. This algorithm utilizes the Power of Two Choices (Po2C) method to strike a balance between better Quality of Service (QoS) and lower costs [1].

In a three-tiered framework, Stavrinides et al. [14] introduced the hybrid fog with a cloud-aware heuristic called Hybrid-EDF for the dynamic scenarios of many basic IoT operations. The scheduling process in our system takes into account the cost of connectivity when transferring data from IoT-level sensors and devices to virtual computers in the fog layer. Because cloud resources are being used, there is a significant financial cost involved.

A cloud-fog computing system that combines cloud nodes that are managed by a fog provider with cloud nodes that are rented from cloud services was described by Pham et al. [15]. To get the most out of the cloud-fog computing system, it is imperative to strategically divide computing workloads among the processing nodes of each tier. The recommended scheduling technique lowers the cost of using cloud resources while ensuring program execution speed. The budget of the fog supplier and the deadline for finishing a workflow are two other constraints that need to be taken into account in order to support the suggested scheme approach. Kabirzadeh et al. [16] suggested a hyper-heuristic method for handling the workflow scheduling problem in a fog computing environment. This study offered a test-and-special rule-based approach.

Tychalas and Karatza [17] presented a dynamic probabilistic load balancing method by expanding on the weighted round-robin algorithm. Using important server metrics to gauge their present level of activity, together with their processing capacity, this strategy assigns probability to available resources.

Subramoney and Nyirenda [10] used the well-known PSO for workflow scheduling to compare cloud and fog-cloud collaboration. Three important elements are taken into account in the provided weighted objective function: cost, makespan, and energy usage. Furthermore, they simulate cloud and cloud-fog systems using the newly created FogWorkflowSim. A comparative evaluation of population-based approaches for process job scheduling in IoT-fog-cloud systems was put out by Subramoney and Nyirenda [18]. In comparison to standard PSO, IPSO offers several advantages. By gradually reducing the inertia weight over iterations, it promotes a balance between exploration and exploitation. This enables the algorithm to efficiently explore the search space early on and then focus on refining promising solutions. As a result, IPSO often achieves faster convergence, better solution quality, and improved adaptability to different problem instances compared to standard PSO.

In order to minimize costs and makespan, Ali et al. [19] proposed a task scheduling strategy based on the Multi-objective Optimization Problem (MOP). This method assigns jobs to fog or cloud devices automatically by utilizing a model that incorporates Discrete Non-dominated Sorting Genetic Algorithm II (DNSGA-II). The burden is efficiently divided between cloud and fog nodes by the model.

Wu et al. [20] demonstrated the effective scheduling strategy used by heterogeneous fog computing systems to reduce power consumption for IOT applications. During the first stages of construction, an integer linear programming technique is applied in an attempt to lower the total energy consumption. The integer linear programming approach looks for key variables that can be changed to reduce energy consumption in a distributed system, as opposed to using them directly for calculation. It is believed that energy is the only goal.

The Budget-Aware Scheduling (BAS) method was proposed by Yadav et al. [21] as a solution to the cloud makespan and cost trade-off for sequencing applications. The method is centered on planning applications according to a timeline to guarantee that they are executed on time, which lowers the costs associated with using cloud resources. Its goal is to improve the use of resources.

Using a PSO technique based on fuzzy resource usage in process scheduling, Farid et al. [22] addressed the Multi-objective Optimization Problem (MOP). As long as dependability limitations are met, the goal is to minimize costs and makespan. Furthermore, the study takes into account the location of job execution as well as the order of data transfer.

To improve the scheduling of scientific work in IoT-cloud-fog systems, Subramoney and Nyirenda [23] devised a technique called multi-swarm particle swarm optimization (MS-PSO), which solves the premature convergence problem of traditional PSO.

G.singh et al. [24] proposed a hybrid algorithm that combines Modified Particle Swarm Optimization (MPSO) with Genetic Algorithm (GA) to address workflow scheduling in a cloud-fog computing environment. The aim is to achieve multi-objective optimization, focusing on enhancing the efficiency and performance of task scheduling by considering multiple objectives such as minimizing makespan, energy consumption, and cost.

Three offloading solutions were introduced by M Gamal et al. [25] with a focus on real-time applications and intended for use in IoT-fog-cloud contexts. When it comes to low latency jobs, LCO works best. Energy efficiency and computationally demanding jobs are given priority by EBO. By optimizing resource utilization, EO seeks to achieve a balance between latency and energy consumption.

3. Offloading and scheduling for workflow

A scientific workflow in a fog-cloud environment leverages the advantages of both fog and cloud computing to handle intricate scientific tasks. This workflow represents a scientific experiment as a sequence of steps, with each step either processing data or conducting a specific analysis in an automated manner. It breaks down the experiment into stages and specifies the order in which they need to be executed. Offloading and task scheduling are two sides of the same coin when it comes to managing tasks in an IoT-fog-cloud environment. Offloading and task scheduling are both strategies used to optimize the performance and efficiency of computing systems, but they address different aspects of this optimization. Offloading refers to the process of transferring tasks or computations from one system or component to another. This is often done to leverage the strengths of different systems or to balance the workload. Task scheduling involves determining the order and allocation of tasks to be executed on resources over time. It ensures that tasks are processed in an efficient and timely manner.

3.1 The concept of a workflow

The concept of a workflow is illustrated using a Directed Acyclic Graph (DAG), denoted as G = (T, E), where T represents vertices corresponding to tasks, labeled from t_1 to tn, and E denotes the edges that represent task dependencies. Each edge signifies data flow between tasks, written as $d_{i,j} =< t_i, t_j > \in E$, where $d_{i,j}$ represents the size of the output data from task t_i to task t_j. Task t_j starts execution only after task t_i has been completed. A task t_i with no predecessor is classified as a starting task, while a task t_j with no successor is classified as an ending task. For example, Fig. 2 demonstrates a workflow with nine tasks. Tasks on the same level (aligned horizontally) can be executed simultaneously, such as tasks t_2, t_3, and t_4, which can run in parallel.

In an IoT-cloud-fog environment, the process of offloading and scheduling a workflow involves distributing tasks across different computing resources, each with its own distinct characteristics. The aim is to optimize workflow execution by reducing three key factors: total completion time (makespan), energy consumption, and overall cost.

3.2 Performance parameters

This setup involves three main categories of computational resources: end devices, fog servers, and cloud servers. These categories include processing and storage capabilities, as well as bandwidth, memory, and power requirements. In the fog and cloud segments, computational resources are represented by virtual machines (VMs). End devices are included because certain minor tasks are more efficiently handled locally due to economic and resource efficiency, rather than being offloaded to fog and cloud servers.

The research centers on a key optimization goal: determining the most effective strategy for task offloading and resource selection to minimize three objectives: cost, makespan, and energy consumption.

3.2.1 Makespan

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

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

Where, inside a specified workflow, ST_{ti} represents the task's start time and FT_{ti} its finish time.

3.2.2 Energy consumption

Idle and active components, denoted as E_{idle} and E_{active}, respectively, are used to determine energy usage [26]. The first one is the energy used when the resource is idle, and the second is the energy used while the task is being executed. The following formula is used to determine the energy used during the idle period [10, 26]:

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

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 $V_{min\ j}$ denoting the minimum voltage for resource j, respectively. $L_{j,k}$ represents the duration of the idle time for $idle_{i,k}$. Therefore, the active energy is determined by.

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

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:

$$TE = E_{active} + E_{idle}. \qquad (4)$$

3.2.3 Cost

This includes the costs associated with computation and communication. All three categories of computing resources have associated costs. On the other hand, communication expenses are not incurred when tasks are completed on the end device. The following is an outline of the computational cost [26] related to using resources to compute r.

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

where pr stands for the cost of processing a unit. The communication cost, which is the cost of data transmission for transferring a task's output of dimension $d_{i,j}$ from the resource that is performing task i to the resource that is assigned to handle job j, is determined by:

$$CC_{i,j} = trc_{i,j} \,.\, d_{i,j}, \qquad (6)$$

In this case, the cost of individual communication between the resource assigned to job j and the resource chosen for task i is denoted as $trc_{i,j}$. If both jobs are completed on the same resource, then $trc_{i,j} = 0$. As a result, the total cost TC for a situation with n tasks and m computational resources can be written as follows:

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

This This study emphasizes minimizing the sum weighted objective function, which is a valuable tool for task scheduling in fog-cloud environments. It allows for scheduling decisions that consider multiple, often competing, objectives, leading to more efficient and practical results. Therefore, the objective function is defined as follow:

$$F(p) = w_1.MS + w_2\,.TE + w_3\,.TC \qquad (8)$$

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. The coefficient weights w_1, w_2, and w_3 represent the degree of importance assigned to each objective. Equal weights are applied in the performance evaluations to ensure that each objective contributes equally, with $w_1 = w_2 = w_3 = 0.2$.

4. Proposed method

Kennedy and Eberhart [27] introduced PSO, a computational technique for solving optimization issues, and IPSO is an extension of PSO. PSO models the collective behavior of individual particles in a swarm, inspired by the social dynamics of fish aggregating or birds soaring. The swarm collectively searches the solution space to discover the optimal solution, with each particle representing a potential solution to the optimization problem. PSO is straightforward, easy to comprehend, and reasonably simple to use. Nevertheless, because of its constant inertia weight, it has an early convergence problem. Thus, multi objective IPSO is proposed to overcome this issue by decreasing weight linearly and dynamically by using a weight that starts high and decreases linearly over time.

4.1 Particle representation

This section explains the particle denoted by "p," which is part of the objective function in Eq. (8), before discussing the rationale behind the proposed IPSO. In this study, tasks within workflows can be executed either at the source (end device), a cloud VM, or a fog VM. End devices do not delegate their tasks to other end devices; they can only assign tasks to cloud and fog resources. Therefore, when scheduling workflows, only a single representative end device is included in the encoding process. Since scheduling tasks in a cloud-fog environment is inherently discrete, natural numbers are used to encode individuals for the proposed IPSO algorithm. Particles represent mappings between tasks and the resources on which they'll be executed. Each particle, denoted as "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 Fig. 3 depicts a workflow with T=9 tasks that need to be mapped to an IoT-cloud-fog environment with R=6 available resources (including 3 cloud VMs, 2 fog VMs, and 1 end device VM). In this scenario, the particle p = {5, 2, 1, 4, 3, 6, 1, 6, 4} 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 6) for that specific task.

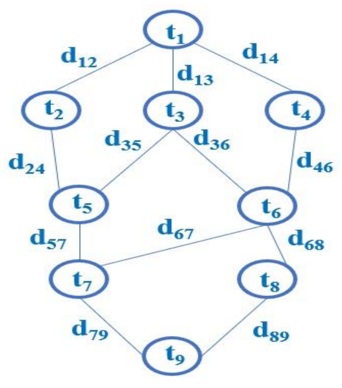


Fig. 2 Workflow of nine tasks example.

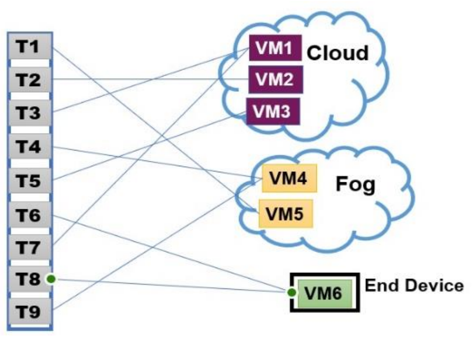


Fig. 3 Workflow scheduling example.

4.2 Multi-objective IPSO for scheduling of workflow

IPSO is an optimization method used in scientific workflow scheduling. It aims to identify the most efficient allocation of tasks to available computing resources. This efficiency is typically measured by minimizing the overall fitness function, also known as a summation of weighted makespan, total cost, and average energy consumption of the entire workflow.

In IPSO, a population of individuals is represented by particles; using the solution space as a guide, they move through it using their existing positions x^k_i and velocities v^k_i for the kth number of iterations. Each particle's quality is evaluated using a predefined fitness function tailored to the specific optimization problem. Each particle's movement is influenced by its own best-known position, pBesti, as well as the best-known global position, gBest, of the entire swarm. This iterative process directs the swarm toward the optimal solution. The flowchart of IPSO is declared in Fig. 4. The stages of the IPSO algorithm are as follows:

1. Initialize IPSO parameters like the number of particles (N), search space boundaries, minimal and maximal inertia weight, learning factors, and maximum iterations (G).
2. Randomly set the initial positions and velocities of the particles within the search space.
3. Begin calculating the value of the fitness function for each particle in the swarm by sum weighted objective function using Eq. (8).
4. For each particle, update its personal best position (pBest) if the current position has a better fitness score.
5. Identify the particle with the best fitness (gBest) in the entire swarm.
6. Do for each iteration the following:
    - ➢ Decrease the inertia weight (ω) linearly from its maximal value (ωbegin) to a minimal (ωend) as:

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

    - ➢ Update each particle's velocity by considering its current velocity, the difference between its position and both pBest/gBest, and the new inertia weight as follows:

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

    - ➢ Update each particle's position by adding its current velocity to its position as follows:

$$x_i^{k+1} = x_i^k + v_i^k \qquad (11)$$

    - ➢ Calculate the fitness of the particles' positions after updating their locations. If a particle's new position has a better fitness value, update its pBest.
    - ➢ Check if the maximum number of iterations has been reached.
    - ➢ If the termination condition is satisfied, exit the loop and return the best solution (the schedule).

In Due to its simplicity, the Socio-Cognitively Inspired standard PSO [10] is used in this work. However, it suffers from premature convergence. To address this, the technique involves discretizing each dimension variable into integers from the fog server set p= {1, 2, …., n}to schedule particle positions, as discussed in the previous subsection. This approach aligns particle positions with task scheduling. Additionally, the weight ω is decreased linearly over iterations, starting high and reducing gradually. This adjustment helps fine-tune the particles' search behavior throughout the optimization process, ensuring continued exploration of the search space and mitigating premature convergence.

In comparison to standard PSO with a fixed weight, IPSO provides a better balance between exploration and exploitation, which can lead to improved schedules by optimizing resource utilization and minimizing execution time. However, a very rapid decrease in ω might restrict exploration too early, potentially limiting the discovery of diverse solutions.

5. Performance evaluation

This section begins with a detailed overview of the workflow models utilized in the study. It thoroughly describes the setup of the simulation environment, including the configuration and parameters used, employing the FogWorkflowSim Toolkit [13]. This toolkit is integral for simulating and analyzing the performance of various workflow models in a fog computing environment. Following the initial setup, the section progresses to present the experimental results derived from the simulations. It includes a comprehensive analysis of these results, highlighting key findings and insights. The discussion delves into the implications of the results, comparing them with theoretical expectations and previous studies, and assessing their impact on the field. This in-depth analysis aims to provide a clear understanding of the performance and effectiveness of the workflow models in the given simulation environment.

5.1 WORKFLOW MODELS

This research employs the established Montage workflow [28] from the field of astronomy to thoroughly assess the effectiveness of the proposed method. The Montage workflow, described by the Pegasus framework [29], utilizes XML files containing DAG representations as input for the simulations. These DAGs meticulously outline the structure of the workflow, including the various tasks, their dependencies, run-times, and required data transfers. This detailed representation allows for accurate simulation and evaluation of the workflow's performance in different computational environments, such as fog computing. Fig. 5 illustrates the workflows in a graphical arrangement, clearly depicting the interconnected tasks and their dependencies. This visual representation helps in understanding the workflow's execution flow and the complex interactions between its components. By employing the Montage workflow, the research provides a thorough assessment of the proposed method's effectiveness in handling real-world, data-intensive applications in astronomy.

The Montage workflow simulates an astronomical application designed to generate custom mosaics of the sky by utilizing several input images. This workflow is particularly relevant in the field of astronomy for its ability to create detailed and accurate composite images from various astronomical surveys. It involves several stages, each contributing to the overall goal of producing high-quality mosaics.

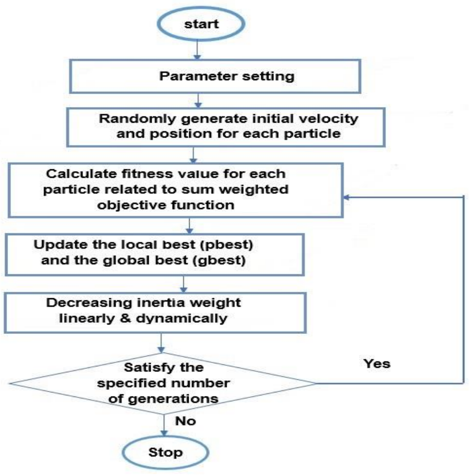

start
Parameter setting
Randomly generate initial velocity and position for each particle
Calculate fitness value for each particle related to sum weighted objective function
Update the local best (pbest) and the global best (gbest)
Decreasing inertia weight linearly & dynamically
Satisfy the specified number of generations
Yes
No
Stop

Fig. 4 The flowchart of IPSO.

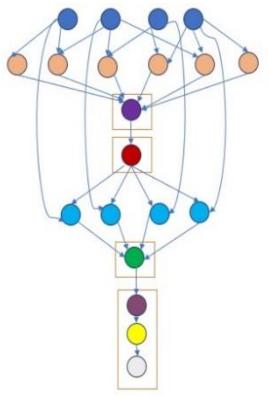


Fig. 5 Structure of Montage workflow.

5.2 Simulation environment

The FogWorkflowSim simulator is run using the Eclipse Java IDE. Simulations are performed on a computer system with a 64-bit Windows 10 operating system, an Intel(R) Core (TM) i7-M 640 @ 2.80 GHz processor, and 6 GB of RAM. Each algorithm is executed with 20 particles.

In IPSO, the learning factors C1 and C2 are set to 2, with the initial and final inertia weights (ωbegin and ωend) set at 0.8 and 0.3, respectively. To assess average performance, each workflow is simulated 10 times. For PSO, the inertia weight is set to 1, with learning factors of 2 for both C1 and C2. The simulation environment consists of 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 detailed parameter configurations, are provided in Table 1.

Table 1 IoT-cloud-fog environment parameter setting [23].

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

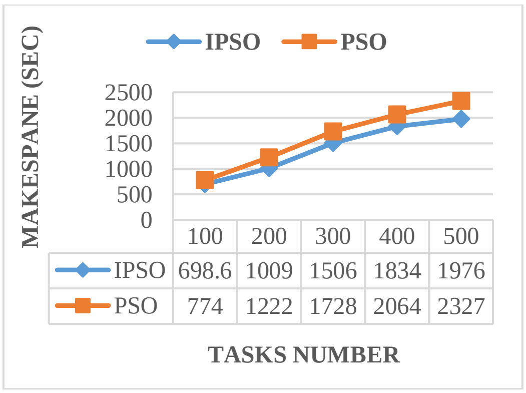
5.3 Simulation results

The performance of the IPSO algorithm is evaluated in comparison to the standard PSO [10] under identical simulated conditions to ensure a fair assessment. The graph in Fig. 6 illustrates the makespan (completion time) for both IPSO and PSO algorithms when applied to the Montage workflow. The x-axis represents the number of tasks in the workflow, ranging from 100 to 500. The y-axis shows the makespan. This figure likely compares how the execution time of these algorithms changes as the size of the workflow increases.

The IPSO algorithm consistently demonstrates superiority over PSO in handling scientific workflows of varying sizes. This is evident in Fig. 6, where IPSO achieves significantly lower makespan (completion time) compared to PSO for all evaluated workflow sizes. When processing 100 tasks, IPSO reduces makespan by an impressive 9.74%. This advantage becomes even more pronounced with larger workflows, reaching reductions of 17.47%, 12.84%, 11.17%, and 15.11% for 200, 300, 400, and 500 tasks, respectively. These findings clearly illustrate the effectiveness of IPSO in optimizing scientific workflow execution compared to PSO.

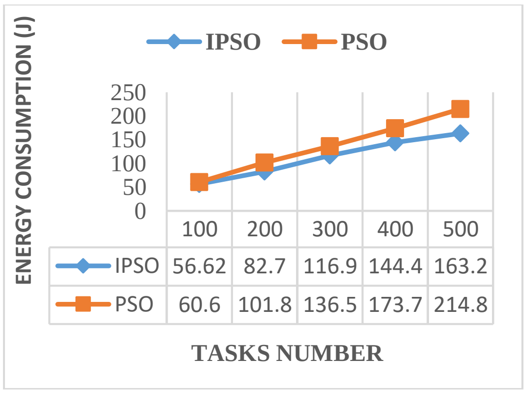
The average energy consumption is displayed in Fig. 7. The number of tasks in the workflow is shown by the x-axis, which ranges from 100 to 500. This average energy consumption is displayed on the y-axis. This figure probably compares how these algorithms' energy consumption varies with workflow size. For scientific processes of different sizes, the IPSO algorithm regularly shows superiority over PSO in terms of energy consumption reduction. This can be seen in Fig. 7, where for all process sizes examined, IPSO achieves noticeably lower energy consumption than PSO. IPSO provides a remarkable 6.56% energy usage reduction when processing 100 tasks. With bigger workflows, this benefit becomes even more noticeable, with savings of 18.78%, 14.39%, 16.85%, and 24.02% for 200, 300, 400, and 500 jobs, respectively. These results imply that, in comparison to PSO, IPSO can greatly increase the energy efficiency of scientific workflow execution.

The IPSO algorithm consistently outperforms PSO in terms of cost-effectiveness for scientific workflows of varying sizes, as evidenced by Fig. 8. IPSO achieves a significant reduction in total cost compared to PSO across all evaluated workflow sizes. When processing 100 tasks, and 200 tasks, IPSO offers a notable 3.54%, and 5.24% cost reduction. This advantage becomes even more diminished with larger workflows, reaching savings of 0.71%, 0.83%, and 1.42% for 300, 400, and 500 tasks, respectively. These findings suggest that IPSO can substantially improve the cost efficiency of scientific workflow execution compared to PSO.



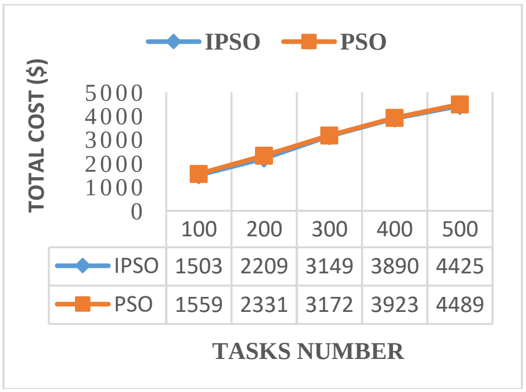
	100	200	300	400	500
IPSO	698.6	1009	1506	1834	1976
PSO	774	1222	1728	2064	2327

Fig. 6 Makespan comparison.



	100	200	300	400	500
IPSO	56.62	82.7	116.9	144.4	163.2
PSO	60.6	101.8	136.5	173.7	214.8

Fig. 7 Energy consumption comparison.



	100	200	300	400	500
IPSO	1503	2209	3149	3890	4425
PSO	1559	2331	3172	3923	4489

Fig. 7 Total cost comparison.

The IPSO time complexity for task scheduling in FogWorkflowSim is O (N^2 * G), where N is the number of tasks in the workflow, and G is the number of iterations performed by the PSO algorithm. This complexity arises from the following factors:

- **Particle evaluation:** Each particle needs to evaluate the fitness function for its current position. This involves calculating the execution time, energy consumption, and other performance metrics for the task schedule represented by the particle. The complexity of this evaluation depends on the specific implementation of the fitness function but is generally linear in the number of tasks (O(N)).
- **Particle update:** The PSO algorithm updates each particle's position based on its velocity and the best positions found so far. This involves simple arithmetic operations on vectors, which have a complexity of O(N).
- **Iteration loop:** The PSO algorithm iterates over a fixed number of iterations (G) to refine the particle positions. Therefore, the overall complexity is multiplied by G.

Therefore, the total complexity of PSO for task scheduling in FogWorkflowSim is O (N^2 * G). This means that the computational time of the algorithm increases quadratically with the number of tasks and linearly with the number of iterations.

6. Conclusion

This study introduces a novel approach called Improved Particle Swarm Optimization (IPSO), designed specifically for managing tasks and scheduling scientific workflows within a complex environment that integrates IoT devices, cloud computing, and fog computing. The main reason for developing IPSO is to address a common issue with traditional PSO: 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 real-time applications where timing, total cost, and energy consumption are crucial. The innovation of IPSO lies in its ability to adjust the inertia weight throughout the process. This allows IPSO to dynamically adapt its behavior, achieving a better balance between the process of searching a wide range of the solution space to discover new and potentially better solutions (exploration) and the process of focusing on refining promising solutions to find the best possible solution within a specific region of the solution space (exploitation). This dynamic approach has the potential to improve both the speed at which IPSO converges on a solution and the overall quality of the solution itself. IPSO surpasses all other methods in minimizing the sum weighted objective function across the Montage workflow with varying numbers of tasks. Additionally, it demonstrates competitive performance in metrics such as makespan, energy consumption, and cost.

In the future, our research aims to further refine IPSO and explore the application of a wider range of algorithms for tackling task offloading and scheduling challenges. We plan to utilize a different workflow. Additionally, to enhance the practicality of IPSO in real-world scenarios, we intend to incorporate constraints such as budget limitations, deadlines, and resource limitations.

References

[1] F. Hoseiny, S. Azizi, and S. Dabiri, "Using the Power of Two Choices for Real-Time Task Scheduling in Fog-Cloud Computing," in Proc. 4th Int. Conf. Smart Cities, Internet of Things and Applications (SCIOT), pp. 18-23, Sept. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9250197 (doi: 10.1109/SCIOT50840.2020.9250197).

[2] R. Tyagi and S. K. Gupta, "A survey on scheduling algorithms for parallel and distributed systems, " in Advances in Intelligent Systems and Computing, vol. 718. Singapore: Springer, pp. 51–64, 2018.

[3] L. Chen, K. Guo, G. Fan, C. Wang, and S. Song, "Resource constrained profit optimization method for task scheduling in edge cloud," IEEE Access, vol. 8, pp. 118638–118652, 2020, doi: 10.1109/ACCESS.2020.3020225.

[4] P. Varshney and Y. Simmhan, "Characterizing Application Scheduling on Edge, Fog, and Cloud Computing Resources," Software: Practice & Experience (SPE), vol. 50, no. 5, pp. 558-595, May 2020, doi: 10.1002/spe.2699.

[5] I. M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. J. Ryan, and K.-K. R. Choo, "An automated task scheduling model using nondominated sorting genetic algorithm II for fog-cloud systems," IEEE Trans. Cloud Comput., vol. 10, no. 4, pp. 2294–2308, Oct. 2020.

[6] M. Akbari, H. Rashidi, and S. H. Alizadeh, ‘‘An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems,’’ Eng. Appl. Artif. Intell., vol. 61, pp. 35–46, May 2017.

[7] F. A. Saif, R. Latip, M. N. Derahman, and A. A. Alwan, ‘‘Hybrid metaheuristic genetic algorithm: Differential evolution algorithms for scientific workflow scheduling in heterogeneous cloud environment,’’ in Proc. Future Technol. Conf., vol. 561, pp. 16–43, 2023.

[8] L. Versluis and A. Iosup, "A Survey of Domains in Workflow Scheduling in Computing Infrastructures: Community and Keyword Analysis, Emerging Trends, and Taxonomies," Future Generation Computer Systems, vol. 123, pp. 156-177, 2021.

[9] G. Chen, J. Qi, Y. Sun, X. Hu, Z. Dong, and Y. Sun, "A Collaborative Scheduling Method for Cloud Computing Heterogeneous Workflows Based on Deep Reinforcement Learning," Future Generation Computer Systems, vol. 141, pp. 284-297, 2023.

[10] D. Subramoney and C. Nyirenda, "PSO-Based Workflow Scheduling: A Comparative Evaluation of Cloud and Cloud-Fog Environments," in Southern Africa Telecommunication Networks and Applications Conference (SATNAC), pp. 258-262, 2021.

[11] J. He and W. Bai, "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), IEEE, pp. 633-638, 2023.

[12] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," IEEE Access, vol. 10, pp. 10031-10061, 2022.

[13] 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.

[14] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," Multimed. Tools Appl., vol. 78, no. 17, pp. 24639–24655, 2019. doi: 10.1007/s11042-019-7353-6.

[15] X. Q. Pham and E. N. Huh, "Towards task scheduling in a cloud-fog computing system," in 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1-4, Oct. 2016.

[16] S. Kabirzadeh, D. Rahbari, and M. Nickray, "A hyper heuristic algorithm for scheduling of fog networks," in 2017 21st Conference of Open Innovations Association (FRUCT), pp. 148-155, Nov. 2017.

[17] D. Tychalas and H. Karatza, "An Advanced Weighted Round Robin Scheduling Algorithm," in ACM International Conference Proceeding Series, pp. 188-191, 2020, doi: 10.1145/3372235.3372242.

[18] D. Subramoney and C. N. Nyirenda, "A comparative evaluation of population-based optimization algorithms for workflow scheduling in cloud-fog environments," in 2020 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, pp. 760–767, 2020.

[19] . M. Ali, K. M. Sallam, N. Moustafa, R. Chakraborty, M. J. Ryan, and K.-K. R. Choo, "An automated task scheduling model using nondominated sorting genetic algorithm II for fog-cloud systems," IEEE Trans. Cloud Comput., vol. 10, no. 4, pp. 2294–2308, Oct. 2020.

[20] C. G. Wu, W. Li, L. Wang, and A. Y. Zomaya, "An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing," Futur. Gener. Comput. Syst., vol. 117, pp. 498–509, 2021. doi: 10.1016/j.future.2020.11.004.

[21] A. M. Yadav, S. C. Sharma, and K. N. Tripathi, "A two-step technique for effective scheduling in cloud-fog computing paradigm, in Advances in Intelligent Systems and Computing," vol. 1086. Singapore: Springer, pp. 367–379, Jan. 2021. [Online]. Available: https://doi.org/10.1007/978-981-15-1275-9_30.

[22] M. Farid, R. Latip, M. Hussin, and N. A. W. A. Hamid, "Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment," IEEE Access, vol. 8, pp. 24309–24322, 2020.

[23] D. Subramoney and C. N. Nyirenda, "Multi-Swarm PSO Algorithm for Workflow Scheduling in Cloud-Fog Environments," IEEE Access, vol. 10, pp. 117199-117214, 2022.

[24] G. Singh and A. K. Chaturvedi, "Hybrid modified particle swarm optimization with genetic algorithm (GA) based workflow scheduling in cloud-fog environment for multi-objective optimization," Cluster Computing, vol. 27, no. 2, pp. 1947-1964, 2024.

[25] M.Gamal, S.Awad, Rehab F. Abdel-Kader,and K. Abd El Salam , "Efficient Offloading and Task Scheduling in Internet of Thingth-Cloud-Fog Environment," International Journal of Electrical & Computer Engineering, vol.14, pp. 4445-4455, 2024. [Online]. Available: http://doi.org/10.11591/ijece.v14i4.pp4445-4455.

[26] 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.

[27] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, Nov. 1995.

[28] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in Proceedings of the 3rd Workshop on Workflows in Support of Large-Scale Science, pp. 1–10, Nov. 2008.

[29] The Pegasus Website. Accessed: Jan. 10, 2022. [Online]. Available: https://pegasus.isi.edu/