A Meta-Heuristics-Based Solution for Multi-Objective Workflow Scheduling in Fog Computing

Gyan Singh, Vivek Dubey
Department of Computer Applications, Government Engineering College, Ajmer, India 305001

***Abstract*—In recent years, there has been a significant increase in the volume of data generated by Internet of Things (IoT) applications, mostly driven by the rapid proliferation of IoT devices and advancements in communication technologies. The conventional cloud computing network was not specifically built to handle such a vast volume of data, leading to several issues, including increased processing time, higher costs, larger bandwidth usage, increased power usage, and communication delays. As a solution, conventional cloud servers have been expanded to additional layers of computing, storage, and network, termed as cloud-fog computing. The cloud-fog computing provides storage, processing, networking, and analytics capabilities in close proximity to IoT devices. The problem of scheduling workflow applications in cloud-fog environments to optimize several conflicting objectives is classified as computationally complex. Particle Swarm Optimization is the widely recognized evolutionary meta-heuristic and is the optimal method for implementing multi-objective solutions because of its user-friendly approach and quick converging capability. Despite its wide acceptance, it does have several drawbacks, such as early convergence and solution stagnation. In order to overcome these limitations, this paper establishes a comprehensive theoretical model to schedule workflow applications for cloud-fog systems. The proposed model employs various competing objectives, such as power usage, overall cost, and makespan. To achieve this, we introduce a novel algorithm, learning enhanced particle swarm optimization (LE-PSO), which incorporates an inverse tangent inertia weight policy and adaptive learning factor methods. The efficiency of the LE-PSO is subsequently assessed by employing an operational data set of scientific workflow applications within a cloudsim-based simulation and validated against GAMPSO, EMMOO, PSO, and GA state-of-the-art approaches. The workflow scheduling, we suggest achieves the substantial decrease in makespan and power usage while maintaining the total cost at an optimal level, in comparison to existing meta-heuristics.**

***Keywords*—Fog computing; DAG; workflow applications; makespan; energy; PSO**

I. Introduction

In recent years, the combination of advancements in Information and Communication Technology and the rapid expansion of Big data has given rise to a new paradigm known as the Internet of Things (IoT). The Internet of Things (IoT) enables the connection of billions of tangible objects via Internet Protocol (IP). The Internet of Things (IoT) has a profound influence on diverse domains. The influence of IoT spans a diverse array of equipment, including sensors, automobiles, potable technology, cameras, patient observation devices, and numerous other applications. Moreover, the Internet of Things (IoT) has enabled the creation of a wide range of services and, applications including automotive communications, residential security, medical tracking, natural calamity prediction, scientific process automation, and roadway congestion management. [1], [2]. In accordance with predictions published by McKinsey Global Institute, the economy of IoT devices and applications will grow from $40 trillion in 2020 to $500 trillion by 2030. However, IoT end devices possess constrained capability in regards to processing, storage, and power capacities, despite generating substantial volumes of data [3]. Conversely, conventional data centers lack the capacity to efficiently handle massive volumes of data [4], [5].

In address to the aforesaid challenges Bonomi [6] proposed the concept of fog computing. Fog computing is paradigm that enhances the capabilities of current Cloud servers by extending them to network periphery, closer to IoT end devices. This improves the data center's capacity to manage workflow tasks which require increased real-time processing with agility. Hence, By establishing extra nodes in the cluster at the network periphery, referred to henceforth as cloud–fog environment.

Typically, The cloud-fog framework comprises several tiers. As shown by Fig. 1, The uppermost tier of architectural structure, known as the cloud tier, consists of multiple interconnected data centers. Optimally, the cloud tier is designed to handle and analyze jobs that need significant processing resources and can tolerate delays in data flow because of its long physical separation from the sources of data generators. The intermediate tier, known as the fog tier, serves a connection among the data centers and Internet of Things devices, usually located nearer to the IoT end-devices. Consisting of multiple machines strategically located at the network's periphery. Fog-tier machines have constrained computing, communication, energy, and data repository capacities. The fog tier machines efficiently handle time-critical operations from end-devices. If there is a task that can tolerate delays but requires a lot of computational power, it should be sent to the data centers. Fog tier nodes are hierarchically arranged to the nodes at the bottom levels. The lowest tier, referred to as the IoT end-device tier, comprises of moveable devices, sensors, cellphones, health monitoring systems, disaster prediction tools, computational science devices, and raw information generation sensors. Submission of tasks from lower tiers to upper tiers is necessary for processing requests. These devices possess extremely limited computing, repository, and energy capacities [6], [7].

Cloud-fog computing framework provides several benefits to address essential challenges. The framework exhibits increased complication and heterogeneity as a result of the various communication networks and processing nodes employed.

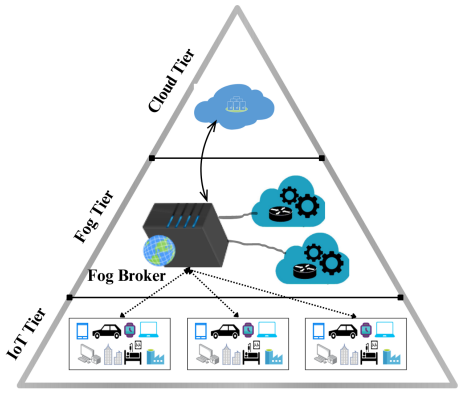


Fig. 1. Cloud-fog framework.

Workflow application scheduling is key challenge for those that need fast runtime, but poses with complex inter-dependent tasks in application and a huge volume of data moving between tasks [8].

Workflows represent the various applications that execute on end devices. The workflow applications are comprised of a set of tasks that have interdependencies. The task dependencies are arranged such that each job must wait for its predecessors to complete before it can be run. Domains such as astrophysics, patients monitoring, natural hazard prediction, computer imaging, and bio-informatics are described as workflow applications. A Directed Acyclic Graph (DAG) is a perfect data structure for representing workflow tasks and the relationships between them efficiently [9]. The aim of optimizing the scheduling of workflow applications is to provide mutually beneficial situations for cloud service providers and clients. This involves considering various parameters and constraints, such as cost, security, energy, deadlines, load balancing, budget, resource utilization, and makespan.

Fog and cloud nodes that are deployed across different geographical locations. Network technologies facilitate the interconnection of computing and storage servers. Enormous heat dissipation solutions are required to maintain essential functional temperatures of cloud data centers. Furthermore, the limited availability of resources in the end-devices tier and fog tier is contingent upon the use of uncertain power sources, like battery storage and sustainable power options. Therefore, the network of the entire cloud-fog system consumes huge amount of energy, with each activity often requiring an average of 15 megawatts. The Communication and information systems sector is responsible for emitting approximately 1.6% of worldwide CO_2 emissions between 2007-2016, in addition to its operational expenses. The predicted rise in this figure is anticipated to reach around 14% between 2016 and 2040 [10]. Consequently, researchers have shown significant interest for the development of sustainable cloud solutions. The challenge of optimizing power consumption in cloud-fog computing has become a significant concern [11]. Dynamic Voltage and Frequency Scaling (DVFS) is a widely used approach for conserving energy. All current processors are equipped with many cores to facilitate the implementation of Dynamic Voltage and Frequency Scaling (DVFS) algorithms. The DVFS technology allows individual CPUs to function at varying voltage and signal frequencies levels. However, a drawback of this technology lies in increase of cost and makespan as well as decreasing power usage. Consequently, this fails to concurrently meet the QoS requirements of both consumers and service providers.

An optimization strategy focused on minimizing the overall runtime of an application might lead to significant energy consumption and resource utilization, hence causing service providers to incur more costs. On the other hand, an algorithm that prioritizes minimizing energy usage can decrease costs for service providers but may result in a longer duration of completing tasks, ultimately causing disagreement among consumers. Hence, employing single-objective techniques to optimize scheduling algorithms is never the most optimal decision. Multiple-objective optimization methods are the most suitable option for achieving a balanced compromise among many optimization objectives. Hence, developing and improving a methodology for scheduling workflow applications in a complex system like cloud-fog, which involves more than objective with competing in nature, is a challenge that falls under the category of NP-hard [12], [13], [14]. Several workflow scheduling techniques have been developed in cloud environments to address various conflicting objectives. Nevertheless, there exist only a limited number of scheduling algorithms that effectively maximize both makespan and cost at the same time. Based on our research, currently, cloud-fog systems have no solution that optimizes the energy usage, cost, and completion time simultaneously. Several well-known evolutionary meta-heuristic methods, such as Genetic Algorithms (GA), Differential Evolution (DE), and, Particle Swarm Optimization (PSO), etc, are considered solutions for optimizing the issue of multi-objective workflow scheduling. most of the above methods focused on optimizing the cost and makespan as the optimization objectives with budget and deadline constraints. Two recent surveys have shown that PSO-based approaches are well-suited and widely used for solving scheduling problems for tasks and workflows with several objectives simultaneously [12][15]. The PSO meta-heuristic algorithms are selected for their benefits such as simple implementation, efficient execution, and rapid convergence. Although PSO has several benefits, it also has certain drawbacks like being trapped in local optima and convergence stagnation. As a result, it may either produce degraded outcomes or could necessitate additional time for computation. In order to address limitations, our research introduces the learning-enhanced particle swarm optimization (LE-PSO) algorithm. This is achieved by incorporating the inverse tangent-based new inertia weight updating and new learning factor method, which aim to strike a balance between the exploitation and exploration capabilities of the PSO methodology. The main work of this paper is listed as follows:

- Develop a theoretical framework to measure performance metrics of multiple-objective scheduling for Workflow applications in the cloud-cog environment that aims to minimize the makespan, cost, and energy usage.
- In order to overcome all of the challenges encountered in standard Particle Swarm Optimization (PSO), we present a new approach called learning enhanced particle swarm optimization (LE-PSO). This algorithm

incorporates a new inverse tangent inertia weight method and a new learning factor method, which aim to enhance both global and local search capabilities.

- Implement the suggested Multi-Objective LE-PSO Workflow scheduling in a cloud-fog environment using the Fogworkflowsim tool [16].
- We analyzed and compared the efficacy of the proposed solution with the four solutions GAMPSO [17], EMMOO [18], Standard PSO, and GA using several real-world scientific workflows. The findings showed a significant reduction in both energy consumption and makespan while not increasing the overall cost.

The subsequent sections of the article are structured in the following manner: Section II provides a comprehensive overview of the current state-of-the-art research on workflow scheduling proposed for cloud fog, fog, and cloud. Section III presents the theoretical framework for scheduling workflows in cloud-fog system. It also introduces the application model. Section IV introduces standard Particle Swarm Optimization (PSO) and the proposed learning enhanced variant of PSO (LE-PSO). Section V provides a comprehensive explanation of how the suggested algorithms can be used to map workflow tasks to a cloud-fog system. Section VI demonstrates the efficacy of the LEPSO and performance evaluation in comparison to existing solutions. Section VII finally finishes by summarizing the contribution of this article and offering insights for further research.

II. Literature Survey

The problem of scheduling in contexts that are both heterogeneous and distributed has been proven to be NP-hard. The challenge becomes more complicated when a workflow application is present and tasks necessitate a predetermined execution order to preserve the inter-dependency among them. The limitation of standard scheduling systems such as HEFT, FCFS, RR, MinMin, etc, is that they only prioritize optimizing execution time as a resource. NP-hard problems necessitate heuristic approaches to generate multi-objective approximate optimal solutions. Evolutionary meta-heuristics are the most commonly used methods for solving scheduling problems in environments such as heterogenous [12]. Various meta-heuristics-based strategies for scheduling workflows have been developed in cloud environments [13], [14], [19], [20], [21], [22]. Referenced algorithms consider either single or multiple objectives such as cost, power usage, load distribution, and completion time, considering constraints of network bandwidth, deadlines of time, and budget. Most research studies on workflow scheduling approaches have been implemented in cloud and distributed environments, with very few articles explicitly concentrating on optimizing various objectives in cloud-fog and fog environments.

In research [23], the discrete form of the Butterfly Optimization meta-heuristics is utilized to optimize workflow scheduling in a mobile-edge computing environment. The authors enhance the algorithm's ability to perform both local and global searches within the solution space by introducing a novel "Levy flight-based" equation. This meta-heuristic approach also focuses on minimizing energy consumption as the primary objective by employing the Dynamic Voltage and Frequency Scaling method. While this method significantly improves data access rates and reduces energy usage, it does not provide any statistical analysis regarding makespan or overall cost.

In study [24], the authors present IKH-EFT: An enhanced technique for workflow scheduling in fog-cloud environments using the krill herd algorithm which presents a novel method that employs the krill herd algorithm to improve efficiency in fog-cloud computing environments. This approach aims to achieve a balance between many goals, such as reducing the time required to complete a task, lowering energy usage and cost, and optimizing customer satisfaction.

In study [25], the Opposition-Based Learning variant of the Harris Hawks Optimization technique is employed to improve the scheduling of workflow applications in a multi-tier fog environment. A predictive model based on the Hidden Markov Model is developed to improve the ratio of missed deadlines and optimize task offloading. Although, this integration is designed to minimize the makespan and optimize the allocation of virtual machines (VMs), however, The findings indicate that the suggested approach is specifically applied to Fog systems, with a primary focus on minimizing only makespan.

In study [3], the work proposed introduces the Scheduling approach to mapping tasks in a cloud-fog environment that integrates two meta-heuristics techniques Genetic Algorithm and Particle Swarm Optimization. This approach seeks to achieve an Optimum trade-off between total cost efficiency and makespan while meeting customer quality satisfaction requirements. Nonetheless, this approach omits energy optimization and has not been tested on the dependent workflow tasks.

In study [26], author presents cuckoo search optimization(COA) algorithm to schedule jobs in smart grids to effectively allocate resources and enhance load balancing. COA is employed to allocate appropriate tasks to Virtual Machines (VMs) with the aim of identifying VMs that are not being fully utilized and VMs that are being excessively utilized. The under-utilized VMs are then powered off to minimize energy usage.

In research [27], the Proposed method integrates the two algorithms: Grasshopper Optimization and Symbiotic Organisms Search. This method was implemented to optimize the energy efficiency in workflow scheduling in fog environment. This integration is facilitated by employing learning automata to determine the most effective algorithm. Additionally, the algorithms utilize Dynamic Voltage and Frequency Scaling techniques to minimize energy usage in fog computing environments, while also managing to maintain an optimal makespan. Although this method successfully lowers energy consumption, the complexity is heightened due to the hybridization of algorithms, and the aspect of overall cost management is not addressed.

In research [28], introduces a novel meta-heuristic algorithm called the Hybrid Particle Whale Optimization Algorithm (PWOA). The PWOA combines two well-established algorithms: Particle Swarm Optimization (PSO) and Whale Optimization Algorithm (WOA). The hybrid approach leverages the advantages of both PSO and WOA, aiming to enhance performance while reducing the limitations inherent in each individual algorithm. The algorithm optimizes the overall

execution time and overall cost for workflow scheduling in a cloud-fog environment. The algorithm omitted energy consumption as a parameter.

In study [29] continuation with PSO and GA combinations studies, this article introduces a novel method by hybridizing the GA and PSO meta-heuristics to improve the optimization of workflow scheduling within a cloud-edge environment. The author harnesses the Genetic Algorithm's two operators: Mutation and selection to increase the randomness in the diversity of the population, while simultaneously implementing the inertia update with the Non-linear method to facilitate a trade-off between global and local searching processes. Although this method shows superior performance in minimizing makespan and cost, it ignores the energy optimization parameter.

In study [30], the author suggests using the Multi-objective Hybrid Dragonfly Algorithm (MHDA) as a hybrid optimization tool to improve task scheduling for Big Data applications in IoT cloud environments. The objective of this algorithm is to minimize the makespan, which refers to the entire time needed to perform a given collection of tasks, while simultaneously maximizing resource use. The scheduling process takes into account key criteria such as makespan, resource utilization, and cost. The main focus of this research is on IoT cloud computing environments, which involve the integration of Internet of Things (IoT) devices with cloud computing resources.

In study [31], the author proposes a list-based dependent task scheduling algorithm for fog computing, which operates in three phases: sorting, prioritization, and selection. In the sorting phase, all independent tasks are organized. In the prioritization phase, tasks are assigned priorities based on their successor tasks. Finally, in the selection phase, task submission decisions are made by balancing global and local search strategies. This approach aims to optimize cost and makespan as its objective functions. However, a limitation of this work is the absence of a meta-heuristic approach; as the number of workflow tasks increases, both makespan and cost also increase.

In study [32], the author introduced a bi-objective workflow scheduling method aimed at optimizing both scheduling reliability and workflow makespan simultaneously in a cloud-fog environment. The model addresses the problem as a multi-criteria challenge, aiming to enhance scheduling reliability while improving the service delivery time ratio for workflow tasks allocated to computing resources. furthermore, it develops a reliability-deadline optimization model, which seeks to optimize application execution time and reliability. This is achieved by adapting a reliability-recursive optimization approach and remapping workflow applications that are not on the critical path. however method neither considers the energy as function objectives nor any meta-heuristics methods employed.

In study [17], a continuation with another PSO-GA-based method introduces a novel workflow scheduling algorithm by hybridizing the Genetic algorithms with the Particle swarm optimization algorithm. The algorithm starts with random initialization of GA and then passes the second half of iteration to PSO. It considers function objectives energy consumption, incurred cost, and makespan. due to the hybridization of GA and PSO, the algorithmic complexity is increased.

In study [18], a multi-objective approach to workflow scheduling is introduced with the goal of minimizing both energy use and makespan while adhering to deadline constraints. The proposed approach operates in two phases. Initially, a task priority queue is generated using Estimated Processing Times to schedule tasks. The subsequent phase employs a DVFS technique to reduce energy consumption without compromising deadline adherence. The idle slots calculated in the initial phase are leveraged in the next phase, eliminating the need for rescheduling. Although this method effectively reduces makespan and consumption of energy, it neither considers cost nor incorporates any meta-heuristic algorithms.

In [33], the author implements a Shortest Job Queue-based job scheduling algorithm for fog environments. The algorithm utilizes the shortest job queue to reduce the application loop delay and network time for submitted applications. It considers factors such as response time, energy consumption, and network usage as a defined objective function. However, the algorithm has not been tested for dependent task submission, and no meta-heuristic approaches have been applied.

The table labeled as Table I provides a summary of the underlying concepts of algorithms, performance measurement metrics, evaluation tools, the context where the algorithm deployed, and challenges/open issues of the literature survey conducted. Prior research in the field has extensively focused on enhancing the efficiency of workflow scheduling in cloud environments. Nevertheless, there is a significant shortage of studies pertaining to fog and fog-cloud. Furthermore, as observed from the literature studies the majority of existing studies primarily concentrate on reducing the time it takes to complete a task and related costs while disregarding the inclusion of energy usage as a key goal. This omission is particularly noteworthy considering the present importance of energy efficiency. In addition, the study shows a preference for using evolving meta-heuristics frameworks to schedule workflows in fog-cloud computing. The primary implementation issue with meta-heuristics methods is the complexity inherent to them. The implementation of meta-heuristics has a tendency to rise the time to complete algorithms, which may not be acceptable for the client's applications that require fast response times. PSO is considered the most appropriate choice for multiple-objective optimization among the many evolutionary meta-heuristics. Nevertheless, it is hampered by certain deficiencies. To improve the applicability of PSO in Workflow scheduling and overcome its drawbacks in cloud-fog, more improvement and refinement are necessary. In order to address the challenges specified in Table I, we have implemented a learning enhanced particle swarm optimization (LE-PSO) for the purpose of scheduling workflows in a cloud-fog environment. This technique takes into account many conflicting objectives, such as Energy, Makepsan, and cost, as the main optimization metrics.

III. System Modeling and Problem Formulation

In this section, We first establish a multi-tier resource model, within which proposed workflow scheduling will be executed. Next, the Dircet Acyclic Graph-based workflow application is modeled with multiple but competing objectives. Then, the problem is mathematically formulated as multiple-objective optimization.

TABLE I. COMPARISON SUMMARY OF RECENT CLOUD-FOG-BASED WORKFLOW SCHEDULING METHODS

Ref.	Underlying Concepts	Performance metrics	Simulator	Environment	Challenges and Open issues
[23]	Discrete version of Butterfly optimization	data access ratio, Energy	iFogSim	IoT-edge	No cost/Makespan mentioned
[24]	Multi-objective krill herd algorithm	Makespan, cost, energy	Matlab	Fog-cloud	complex to implement
[25]	Harris Hawks Optimization and Hidden Markov Model	task offloading,missed deadlines, makespan	iFogsim	Fog	No enrgy and cost, No complex environment
[3]	Hybrids PSO with GA	total cost, Makespan	iFogsim	Cloud-Fog	No Energy optimized, No workflow task evaluated
[26]	Cuckoo optimization algorithm	Energy consumtion, response time	cloudsim	cloud-fog	no task dependency,only energy
[27]	Integrate Symbiotic Organisms with Grasshopper Optimization	Energy	iFogsim	Fog	No muil-tier, increased complexity,only energy
[28]	Hybrids PSO with Whale Optimization	Execution Cost, Execution Time	WorkflowSim	cloud-fog	Energy not considered,increased complexity
[29]	Hybrids Non-linear PSO with GA	Total Execution time, total cost	WorkflowSim	Cloud edge	only cost,execution accounted for
[30]	Multi-objective Hybrid Dragonfly Algorithm	Cost, makespan, resource utilization	cloudsim	Cloud-IoT	No energy considered
[31]	list-based task scheduling	Cost, makespan	iFogsim	fog	Non-meta-heuristics employed,under performed in higher load
[32]	simple heuristics based	reliability,makespan	ifogsim	cloud-Fog	no meta-heuristics, No energy parameter
[17]	Hybridization of PSO with GA	energy consumption, cost, Makespan	Workflowsim	cloud-Fog	complexity increased
[18]	DVFS with Sorted priority queue	makespn,energy	MATLAB	cloud-Fog	Meta-heuristics not used, No cost
[33]	Shortest Job Queue-based	energy, response time, network	iFogsim	fog	No dependent tasks, no meta-heuristics

A. Resource Model

The network architecture of the cloud-fog system is structured into three separate tiers: cloud tier, fog tier, and IoT end devices tier. The Fog Broker node is a specialized server node in the fog tier, as depicted in Fig. 1.

The broker node is responsible to coordinate and facilitate communication between three tiers in order to efficiently allocate tasks to appropriate resources, whether they are located in the cloud or fog. The broker node works depending on optimization goals specified by the service provider.

In the proposed research, we create a heterogeneous environment comprising three distinct categories of resources. These include cloud nodes located in reserved high-capacity storage and computing host machines that communicate via a Wide Area Network within the cloud tier; fog nodes that are interconnected through Ethernet in the fog tier; and IoT devices that possess very constrained processing, storage, and communication capabilities, connect to nearby fog nodes over the mobile network, forming end-devices tier. We denote C as a set of virtual machines provisioned on cloud tier, defined as $C = vm_1, vm_2, vm_3, \ldots, vm_c$. Similarly, F denotes the VMs deployed on fog tier, where $F = vm_1, vm_2, vm_3, \ldots, vm_f$. The VMs generated on end-user devices are primarily intended for managing minor tasks. Tasks that need higher computing and storage requirements will be offloaded to the upper tiers. Consequently, resources available on end devices are considered to have minimal significance in terms of performance. $V = C \cup F = vm_1, vm_2, vm_3, \ldots, vm_m$ are the sum of VMs provisioned at cloud-fog and registered at Fog broker node, reflecting their distinct technical specifications.

B. Application Model

Workload applications submitted to cloud-fog will be modeled by direct-acyclic-graph, denoted as $G(T, E)$, Where T is a set of tasks and E is the communication link between tasks. E represented a set of one-way inter-dependencies among tasks. Every task $\tau_s \in T$ in set T have thier size defined by *size(τ_s)*.Every edge $\epsilon_{ij} = \langle \tau_i, \tau_j \rangle \in E$ signifies a dependency of flow or control between tasks τ_i and τ_j. This indicates that τ_i must be finished before τ_j can begin, thereby establishing τ_i as a predecessor of τ_j due to the precedence constraints.The collection of all tasks that precede a given task is represented as *pred(τ_s)*, while the set of tasks that follow is represented as *succ(τ_s)*. The edge ϵ_{ij} is associated with communication weight cv_{ij} with a value zero or greater, representing the amount of data transferred from task τ_i to task τ_j. In cases where a task lacks a predecessor or successor, pseudo task τ_e / τ_x equals zero is introduced into the graph. As shown in Fig. 2, We have created an example to illustrate workload application DAG. It demonstrates how tasks are organized across seven distinct levels. In the third level, tasks τ_2, τ_3, and τ_4 are designed to execute concurrently.

C. Problem Formulation

1) Makespan objective: Total time needed to complete a workload application, measured from the starting time of the initial task, τ_e, to the finishing time of the final task, τ_x, is referred as Makepan. For our proposed model, two distinct categories of VMs have been created: c number of virtual machines in cloud tier and f number of virtual machines in fog tier. Thus, The specialized node fog broker is registered with the sum of VMs ($m = c \cup f$) at the specific moment. Submitted tasks of workload are executed on cloud virtual machines or fog virtual machines depending on the selected offloading strategy by the fog broker to reduce the makespan for the entire workload. When a task τ_s of workload is assigned to a VM deployed at node l, the execution time for τ_s is represented as:

$$ET_{\tau_s^l} = \frac{size(\tau_s)}{PC^l} \quad (1)$$

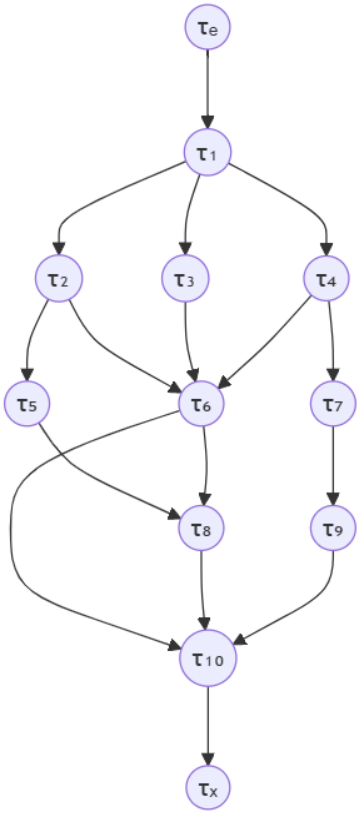


Fig. 2. A DAG-based workload illustration.

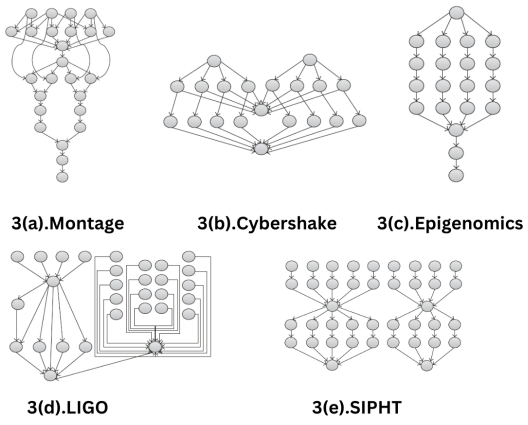


Fig. 3. Structure of various workloads.

Where $ET_{\tau_s^l}$ represents the time required to execute task τ_s on the l^{th} physical machine, $size(\tau_s)$ denotes the size of the task, PC stands for the processing capacity for l^{th} physical machine.

Consider $DT(\epsilon_{ij}^l)$ as the data transfer time between tasks τ_i and τ_j, which can be calculated as follows:

$$DT(\epsilon_{ij}^l) = \frac{size(cv_{ij})}{B^l} \tag{2}$$

$size(cv_{ij})$ represents the volume of data exchanged between tasks τ_i and τ_j, while B denotes the bandwidth available between the physical machine.

The start time ST_{τ_s} and finish time FT_{τ_s} for task τ_s can be determined using the following formula:

$$ST_{\tau_s} = \max(FT_{\tau_p} + DT(e_{ps}^l), \tau_p \in pred(\tau_s)) \tag{3}$$

The set $pred(\tau_s)$ includes all the preceding tasks of task τ_s.

$$FT_{\tau_s} = ST_{\tau_s} + ET_{\tau_s^l} \tag{4}$$

Therefore, the overall makespan of the workload is:-

$$MS = maximum(FT_{\tau_s} - minimum(ST_{\tau_s}), \tau_s \in pred(\tau_s))) \tag{5}$$

2) Cost objective: The service architecture of cloud fog model operates on a business-driven on-demand prices, where pricing structures for their solutions are established by service providers. In this cost model, total expenses are derived from two resource types: computing resources and data transfer communication resources. Both cloud and fog machines are accounted for to determine the total cost, whereas the costs related to end devices are assumed to be minimal.

The computational cost for executing a task τ_s at the physical machine l is determined by:

$$CC_{\tau_s^l} = UC^l * (FT_{\tau_s} - ST_{\tau_s}) \tag{6}$$

CC represents the computational cost, UC^l denotes the unit price on physical machine l.

The cost of communication between tasks τ_i and τ_j will be determined using the following calculation:

$$CD(\epsilon_{ij}^l) = UD^l * DT(\epsilon_{ij}^l) \tag{7}$$

$CD(\epsilon_{ij}^l)$ represents the cost of communication for transferring data between task τ_i and task τ_j, UD^l denotes the unit price. When tasks τ_i and τ_j are processed on the same physical machine, the amount of data transfer $DT(\epsilon_{ij}^l)$ is equal to 0.

Consequently, the sum of cost is determined by:

$$TC = \sum_{l=1}^{m} \sum_{s=1}^{n} CC_{\tau_s^l} + \sum_{l=1}^{m} CD(\epsilon_{ij}^l) \tag{8}$$

Here, m denotes the VM counts, and n denotes the task count.

3) Energy objective: We have derived energy objectives from the framework outlined in [34], which divides energy consumption into two key parts: static and dynamic consumption of energy. The dynamic part of consumed energy denoted as ED, is particularly significant as it leads to higher energy usage during the execution of workload tasks.

$$ED = \sum_{s=1}^{n} C * f_s * {v_s}^2 * (FT_{\tau_s} - ST_{\tau_s}) \tag{9}$$

Here C represents the constant value as a capacitive load, v denotes the supply voltage, where f is the frequency of the CPU where executing the task τ_s.

ES denotes the static part of energy utilized, which functions at the lowest possible frequency with the minimum voltage supply required to keep the system operating. While the system is idle, it carries out crucial functions like functioning core circuits, supporting the continuous working of RAM and input/output activities, and clock running.

$$ES = \sum_{l=1}^{m} \sum_{idle_{l,k} \in IDLE_{l,k}} C * f_{l,mn} * v_{l,mn}^2 * S_{l,k} \tag{10}$$

Where $IDLE_{l,k}$ represents the collection of all idle time slots available on physical machine l, characterized by the minimum frequency $f_{l,mn}$ and minimum supply voltage $v_{l,mn}$, additionally, $S_{l,k}$ denotes the slot of idle time utilized by l.

The sum of utilized energy for a given workload is determined by:-

$$TE = ED + ES \tag{11}$$

D. Fitness Metric

Most evolutionary techniques begin with a randomly initialized population. In Evolutionary meta-heuristics methods, the initial phase of population is distributed inherently stochastic.To minimize skews in various objectives, a normalization process is applied by min-max scaling technique. This process involves following steps for normalizing all three objectives:

$$MS_{adj} = \frac{MS_i - MS_{mn}}{MS_{mx} - MS_{mn}} \tag{12}$$

$$TC_{adj} = \frac{TC_i - TC_{mn}}{TC_{mx} - TC_{mn}} \tag{13}$$

$$TE_{adj} = \frac{TE_i - TE_{mn}}{TE_{mx} - TE_{mn}} \tag{14}$$

In this context, "mn" and "mx" represent the lowest and highest fitness values. while i signifies the iteration number as the solution evolved. The "adj" refers to the normalized fitness value.

Following this, the fitness value of an objective for the mapping of task-to-virtual machine schedule p is determined by employing the normalized objectives, as defined by:-

$$f(p) = \alpha.MS_{adj} + \beta.TC_{adj} + \gamma.TE_{adj} \tag{15}$$

Here α, β, and γ represent weighted constants that satisfy the condition $(\alpha + \beta + \gamma = 1)$.Service providers have the flexibility to adjust the value of the constant based on their specific service requirements and preferences. In this case, constants α,β, and γ are selected in the ratio (0.3:0.3:0.4), reflecting that cost and makespan are given equal importance, while slightly higher importance is given to energy.

IV. LEPSO

This section begins by presenting the core principles of the standard particle swarm optimization algorithm, PSO is extensively used to optimize a wide range of complex problems with multi-objective. Following this, we explore an enhanced version specifically designed for workflow scheduling in cloud-fog systems, with a focus on managing conflicting objectives.

A. Standard PSO

Standard PSO, developed by "Kennedy and Eberhart" [35], is a multi-objective, evolutionary, stochastic optimization technique. It has gained popularity for its simplicity and quick convergence, making it a widely recognized algorithm. PSO is particularly useful for tackling complex challenges such as scheduling tasks of workflows in heterogeneous computing systems [12]. The core idea of the algorithm involves particles that search for the best solutions by coordination among particles, and exchanging information between particles. PSO was inspired by the navigation and foraging behaviors observed in schools of fishes. Consider K particles involved in searching within a D-dimensional search space. Each particle, denoted as p_k (where $k = 1, 2, \ldots, K$), is defined by its position $x_k = (x_{k1}, x_{k2}, \ldots, x_{kD})$, velocity $v_k = (v_{k1}, v_{k2}, \ldots, v_{kD})$ within the landscape. For the i^{th} iteration during the evolution process, the velocity of the k^{th} particle will be modified as follows:

$$v_k^i = \omega^i \cdot v_k^{i-1} + c_1 \cdot r_1 \cdot (pBest_k - x_k^{i-1}) + c_2 \cdot r_2 \cdot (gBest - x_k^{i-1}) \tag{16}$$

The equation defines three components: inertia weight, particle self-learning component, and particle social-learning component. The term $pBest$ denotes the best position achieved by the i^{th} particle during the iteration, while $gBest$ represents the optimal position across the entire search landscape. The variables r_1 and r_2 are random values between 0 and 1. The inertia weight ω and the self-learning (c_1) and social-learning (c_2) factors regulate the problem's exploration and exploitation capabilities within the search landscape.

For the k^{th} iteration of evolution, each particle's position will be modified as follows:

$$x_k^i = x_k^{i-1} + v_k^i \tag{17}$$

In standard PSO, the inertia weight is a crucial element that influences both local as well as global exploration capabilities throughout the process of evolution. It is computed using a linear decreasing method as follows:

$$\omega^i = w^{mx} - \frac{(w^{mx} - w^{mn}) * i}{T_{mx}} \tag{18}$$

Here, w^{mx} and w^{mn} denote the maximum and minimum inertia weights, respectively, while T_{mx} represents the maximum number of iterations, as specified in Table II. A linear decrease in inertia weight means that the global search capability is broader in the early iterations, while the local search capability becomes more focused later on. In various standard benchmarks, $c1$ and $c2$ are uniformly assigned a value of 2 to ensure a proper trade-off between global and local search abilities.

B. Learning Enhanced PSO

The standard Particle Swarm Optimization (PSO) method is straightforward to implement and excels in global search across multiple objectives. However, when applied to workflow scheduling in complex environments, it often yields suboptimal results due to its tendency to get stuck in local optima, which disrupts the balance between local and global search capabilities. Key parameters influencing search performance include the learning factors(c1,c2) and inertia weight(ω). To overcome these issues, dynamic mechanisms for both the learning factor and inertia weight are employed.

1) Enhanced learning factor: c1 and c2 are crucial for managing the balance between global and local search abilities. Both factors are usually assigned constant values derived from past benchmarking. The information given outlines various scenarios depending on c1 and c2.

- When both c1 and c2 are set to 2, the setup seeks to strike a balance between global and local search processes. Nevertheless, this is crucial to note that every iteration in the Particle Swarm Optimization (PSO) process does not benefit from the self-learning ability of previous iterations.
- If $c1 = 0$ and $c2 = 2$, the system does not utilize learning solutions and rapidly converges on local search areas, often yielding sub-optimal outcomes. This scenario suggests less diversity in population and less exploration within the landscape.
- When $c1 = 2$ and $c2 = 0$, particles do not exchange information. This setup prevents the algorithm from converging and reduces its efficiency, as there is no collaborative interaction among the particles.

These findings highlight the need to incorporate a new learning factor approach in PSO to balance global and local search abilities effectively to achieve better optimization. Consequently, we propose adaptive learning factors, c1, and c2, which are computed during each evolution as follows:

$$c_1^i = c^{mn} + \frac{(c^{mx} - c^{mn})(T_{mx} - i)}{T_{mx}} \tag{19}$$

$$c_2^i = c^{mn} + \frac{(c^{mx} - c^{mn})(i)}{T_{mx}} \tag{20}$$

Here, [c^{mx} , c^{mn}] denote the range defined in Table II, T_{mx} maximum number of iteration , i represents current iteration. In the initial phases of this enhanced approach's iteration, focus is on minimizing the self-learning of particles while amplifying social learning. This modification improves global search ability, thereby exploring larger area of landscape. As the process moves into subsequent phases, the approach shifts: the self-learning aspect is elevated, and social learning aspect is reduced. This alteration is aimed to strengthen local search, thereby accelerating convergence toward the optimal global solution.

2) Inertia weight: The self-learning and social-learning term as defined by Eq. (16) is solely accountable for gradually converging the fitness value toward the zero throughout the iterations. This behavior of the equation is governed by the inertia weight ω. If the decreasing rate of w is so fast from ω^{mx} to ω^{mn}, premature convergence may occur, resulting in suboptimal outcomes. If the fitness of the swarm remains immutable for an extended period, there is a high likelihood of encountering a stagnant situation, potentially leading to the degradation of algorithm performance. It is crucial to attentively adjust the inertia weight to address the challenges of stagnating particles and early converging to sub-optimal solutions Even Applying linear decrease in ω as defined in Eq. (18) may lead to an imbalance between pBest and gBest, resulting in premature convergence or stagnation. In order to resolve this problem, we utilize a method with dynamic inertia weight that applies a non-linear inverse tangent decreasing function, computed as defined below.

$$\omega^i = \omega^{mn} + (\omega^{mx} - \omega^{mn}) \times \frac{\arctan\left(a\left(\frac{i - \frac{T_{mx}}{2}}{T_{mx}}\right)\right)}{\arctan\left(\frac{a}{2}\right)} \tag{21}$$

exploration and exploitation are controlled by a=10, which affects how quickly the inertia weight transitions from higher to lower values, becomes dominant as the arctan function flattens out, leading to more stable convergence.

By utilizing non-linear inverse tangent inertia weight and, enhanced learning factors, the modified velocity equation is calculated as follows:

$$v_k^i = \omega^i \cdot v_k^{i-1} + c_1^i \cdot r_1 \cdot (pBest_k - x_k^{i-1}) + c_2^i \cdot r_2 \cdot (gBest - x_k^{i-1}) \tag{22}$$

V. Proposed LEPSO Workflow Scheduling Algorithm

We have implemented cloud-fog workflow scheduling algorithms by utilizing our LEPSO meta-heuristics in this section. Prior to implementing the algorithm, a mapping model is created to correlate the different terminologies utilized in LEPSO with elements of cloud-fog systems.

A. Mapping PSO Particles to Cloud-Fog Resource

A particle in the PSO search space represents a potential solution, making it essential to design particles that can yield optimal results for workflow scheduling. Given that workflow

scheduling is inherently a discrete problem, natural numbers are used to represent particles. In a cloud-fog environment, 'm' virtual resources are provisioned across three categories: virtual machines deployed on the cloud, fog, and IoT end-devices. A set of 'n' tasks in the workload is then mapped to these provisioned virtual resources. The workload's task count determines the dimension 'D' of the particle. Every particle within the PSO population is mapped to a vector consisting of natural numbers. In this vector, each index represents a specific Task ID, while the value at that position denotes VM ID to which the corresponding task is allocated for execution

Consider, for example, a scenario depicted in Fig. 2, where a workload consists of tasks $n = 10$ and there are virtual machines $m = 5$ deployed: 2 in the cloud, 2 in the fog layer, and 1 on IoT end devices. The function of IoT end-devices VM is only to push the workload application to fog broker. The mapping of tasks to virtual machines can be represented by a particle 'P', which is expressed as:

3	3	4	1	2	4	2	4	1	5

Particle 'P' as a vector

The above Task-VM schedule represents allocating every task to map on the designated virtual machine. i.e. , Task1 is allocated on VM3, Task2 is also allocated to VM3, Task3 goes to VM4, and so forth.

B. Implementation of LEPSO Workflow Scheduling

By employing a particle encoding method and the objective fitness function defined in Eq. (15), the algorithm is executed as detailed in Algorithm 1. The input to the algorithm consists of a Directed Acyclic Graph workload G containing a set of tasks $T = \{\tau_1, \tau_2, \ldots, \tau_n\}$ and a collection of virtual machines $V = \{vm_1, vm_2, vm_3, \ldots, vm_m\}$. The output generated is the optimized task-to-VM mapping schedule (gBest).

1) Swarm initialization: Algorithm 1 begins by configuring various parameters of the LEPSO algorithm as detailed in Table II, including the maximum number of iterations (T_{mx}) and the maximum particles participated(K), as described in algorithm's input section. Following this, in lines 3 to 9, The particles in LEPSO begin with randomly assigned positions and velocities. Every particle retains its most optimal known position, referred to as pBest, which denotes a possible task-to-VM schedule. The best overall solution, known as gBest, is identified from among all pBest solutions by evaluating the fitness value determined using Eq. (15).

2) Swarm movement and evolution: Once the population is randomly initialized and all parameters are set, the particle evolution process commences and continues until the maximum number of iterations, T_{mx}, is reached. Throughout each iteration (as detailed in lines 10-23), particles undergo evolution, gradually improving their fitness values. The evolution process is governed by the outer loop, where in each iteration, $c1$, $c2$, and ω are modified as per the respective Eq. (19) to (21), in lines 11 to 13.

Inside the inner loop, ϵ and x for every p are modified by Eq. (22) and (17). Once the inner loop concludes, the global best schedule $gBest$ is identified from all personal best schedules $pBest$ by calculating associated fitness metrics by Eq. (15), as indicated in lines 18 to 21. After reaching the end of T_{mx} iterations, the final gBest solution is returned.

Algorithm 1 LEPSO Algorithm

Input: Workflow task set $\{\tau_1, \tau_2, \ldots, \tau_n\}$, Virtual Machines (VMs) in cloud-fog $\{vm_1, vm_2, vm_3, \ldots, vm_m\}$
Output: Optimal Task-VM mapping $gBest$

1: Initialize T_{mx}, K, $r1$, $r2$, a, b, c, c_{mn}, c_{mx}, ω_{mn}, ω_{mx} as per Table-II.
2: Set initial $gBest$ fitness: $f(gBest) \leftarrow 1.0$
3: **for** k from 1 to K **do**
4: Randomly initialize $pBest_k$, v_k, and x_k.
5: **if** $f(pBest_k) < f(gBest)$ **then** (calculated using equation 15)
6: Update $f(gBest)$: $f(gBest) \leftarrow f(pBest_k)$
7: Update $gBest$: $gBest \leftarrow pBest_k$
8: **end if**
9: **end for**
10: **for** i from 1 to T_{mx} **do**
11: Calculate c_1^i using equation (19)
12: Calculate c_2^i using equation (20)
13: Calculate ω^i using equation (21)
14: **for** k from 1 to K **do**
15: Update v_k^i according to equation (22)
16: Update x_k^i according to equation (17)
17: Refresh schedule $pBest_k^i$
18: **if** $f(pBest_k^i) < f(gBest)$ **then** (calculated using equation 15)
19: Update $f(gBest)$: $f(gBest) \leftarrow f(pBest_k^i)$
20: Update $gBest$: $gBest \leftarrow pBest_k^i$
21: **end if**
22: **end for**
23: **end for**
24: **return** $gBest$

TABLE II. LEPSO ALGORITHMS SETTING

Number of particles(K)	100
Number of Evolutions(T_{mx})	100
Repeated experiments	100
learning factors ($c_{mn} - c_{mx}$)	0.5 to 2.5
r1 and r2	0 to 1
inertia weight[$\omega_{mx} - w_{mn}$]	0.95 to 0.3
Cost weight (β)	0.3
Makespan weight (α)	0.3
Energy weight (γ)	0.4

Optimizing the scheduling of workflow applications in a complex system is particularly challenging because of NP-hardness, which inherently involves balancing algorithmic complexities with optimization goals. Consequently, we analyze and compare the efficiency of LEPSO through a series of experiments conducted using a simulator in the subsequent section.

VI. Performances Analysis

The performance of the proposed LEPSO algorithm was evaluated by implementing it using Fogworkflowsim [16]. Fogworkflowsim extends Workflowsim [36] by adding an additional set of layers: fog, IoT end-devices layers. This platform enables the design, simulating, and evaluation of workflow performance in cloud fog. To verify outcomes, we compared the LEPSO algorithm's performance against the recent approaches presented in GAMPSO [17], EMMOO [18] as well as against standard PSO and traditional GA.

A. Scientific DAG Workflows

The effectiveness of many workflow scheduling algorithms is often evaluated using either randomly generated workflow datasets or real-world scientific workflow datasets. In this study, we utilize the latter. These datasets are represented as XML text documents organized in a directed- acyclic-graph format provided by the Pegasus framework [37]. These text files contain details such as the number of tasks, their execution times, task sizes, the size of outputs transferred between tasks, and dependencies in parent-child relationships. Fig. 3 illustrates the structure of these five scientific workflows.

Montage workflows, developed at NASA, are designed for creating custom mosaics of the sky by arranging multiple input images together [38]. CyberShake, on the other hand, is a workflow created by the Southern California Earthquake Center, which aims to predict earthquake risks within specific areas. Epigenomics workflows automate various processes involved in gene mapping. The LIGO dataset is employed to examine gravity waves produced by the gradual approach and eventual merger of various dense bodies. Lastly, the Sipht dataset is used in the field of Bio-informatics to automate the search for small untranslated RNAs (sRNAs) during bacterial replication.

B. Execution Environment

To assess the effectiveness of the LEPSO algorithm, comprehensive simulations experiments were performed using FogWorkflowsim. The simulator ran on a 64-bit Windows 10 operating system with the following device configurations: Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz, RAM 8 GB. The efficiency of the LEPSO algorithm is evaluated with the Montage workload application dataset, as characteristics listed in Table IV.

To simulate on FogWorkflowsim, five virtual machines (VMs) were allocated to the cloud tier, 5 to the fog tier, and 5 to end-devices. Additional parameters, such as processing and communication capacity, energy consumption, and costs, were defined as per Table III. The parameters listed in Table III were consistently used throughout the execution of the LEPSO algorithm.

The algorithm began by initializing a population of 100 particles. Each particle p underwent 100 iterations of evolution, during which its associated parameters x, υ, ω, c1, c2, and balance coefficients were updated according to the values specified in Table II. For each scenario, the simulation is conducted 100 times to calculate the average outcomes.

For performance evaluation, five scenarios with varying workflow task sizes and dependencies were selected. To measure makespan, cost, and energy consumption in heterogeneous workload submissions, different workloads are selected, ranging from smaller sets containing 100 tasks to larger sets with up to 500 tasks. These Montage workload scenarios are characterized by various complex characteristics such as dependent tasks, task size, input/output file size, and number of tasks, as outlined in Table IV.

TABLE III. Parameters of Cloud-Fog Simulator

Parameters	End-Device	Fog	Cloud
Host	5	5	5
Virtual machine	5	5	5
Computing Speed	1K	2K to 4K	5K to 9K
Computing cost ($)	0	0.2 to 0.5	0.5 to 0.9
Communication cost ($)	0	0.1	0.2
Bandwidth (Mbps)	100	200	100
Static Power (mW)	30	30	1328
Dynamic Power (mW)	700	700	1648

TABLE IV. Characteristics of Various DAG Scenario

Scenario	Nodes	Edges	Input Data(MB)	Output Data(MB)	run time(sec)
1	100	252	15.11	4.37	10.24
2	200	518	15.14	3.85	10.52
3	300	786	15.35	3.87	10.68
4	400	633	11.45	3.62	11.26
5	500	833	12.49	2.9	11.01

TABLE V. The Results of Experiment

Scenario	Algorithm	Makespan (Sec.)	Cost ($)	Energy (joules)
Scenario 1	GAMPSO	350.11	508.17	180
	EMMOO	1,045.45	NA	1,875.00
	PSO	472.48	513.76	62.97
	GA	443.88	512.96	233.93
	LEPSO	**359.21**	**497.04**	**22.80**
Scenario 2	GAMPSO	590.18	721.45	201.63
	EMMOO	1,500.00	NA	2,812.50
	PSO	704.82	770.46	110.12
	GA	793.74	827.34	398.17
	LEPSO	**530.85**	**771.86**	**63.07**
Scenario 3	GAMPSO	901.23	930.56	401.48
	EMMOO	1,909.09	NA	4,687.50
	PSO	1109.69	1103.12	157.03
	GA	923.37	940.92	653.91
	LEPSO	**800.21**	**998.48**	**172.28**
Scenario 4	GAMPSO	385.14	798.29	502.23
	EMMOO	2,318.18	NA	7,500.00
	PSO	611.69	789.05	145.17
	GA	390.84	749.66	755.25
	LEPSO	**380.58**	**748.24**	**135.57**
Scenario 5	GAMPSO	380.18	899.38	600.12
	EMMOO	2,500.00	NA	11562.5
	PSO	694.84	900.97	200.85
	GA	369.80	881.71	1,098.13
	LEPSO	**360.47**	**870.39**	**92.49**

TABLE VI. MEAN RESULT OF VARIOUS SCENARIO

Algorithm	Makespan (Sec.)	Cost($)	Energy (J)
GAMSPO	521.39	771.56	377.09
EMMOO	1,854.54	NA	5,687.50
PSO	718.70	815.47	135.23
GA	584.33	782.52	627.88
LEPSO	**486.26**	**777.20**	**97.24**

TABLE VII. AVERAGE ALGORITHMIC RUN TIME(SEC.)

Algo/ scenario	one	two	three	four	five
GAMPSO	20.34	60.12	301.23	330.94	401.37
PSO	13.41	41.14	94.50	109.77	196.65
GA	39.87	132.18	477.09	440.94	724.17
LEPSO	**18.37**	**45.52**	**195.00**	**228.00**	**206.00**

C. Discussions

To assess and compare the performance of the proposed LEPSO algorithm with the GAMPSO [17], EMMOO [18], standard PSO, and GA, we executed five different scenarios, each scenario with different characteristics as outlined in Table IV. The outcomes for all three performance metrics were recorded and analyzed, as shown in Table V.

For each scenario, we incrementally increased the number of tasks within the workflow. Among the five scenarios, the first has the smallest problem space, whereas the third has the largest. The size of the problem space is influenced not only by the number of tasks but also by other factors, such as the task dependencies, task size, input file size, output file size, and additional characteristics described in Table IV.

The results show that across all scenarios, the proposed LEPSO significantly improves makespan and energy consumption compared to GAMSPSO, EMMOO, standard PSO and GA, with cost also showing slight improvements as mentioned in mean result Table VI. Specifically, in scenarios one and two, there is a minor improvement in cost, but a substantial enhancement in makespan and energy. In the third scenario, which is considered the worst case, a notable decrease in energy consumption and makespan is observed, although total cost is marginally impacted due to increased communication costs resulting from a higher number of edges between cloud and fog nodes. The last two scenarios also demonstrate major improvements in energy consumption and makespan, with minimal enhancement in cost.

It is evident from Table VII that the problem with GA lies in its tendency to create unnecessary population diversity, leading to slow convergence and degraded performance. On the other hand, standard PSO converges quickly but often gets trapped in local optima, and fitness value stagnates, resulting in lower-quality outcomes. GAMPSO performs better results but creates unnecessary algorithmic complexity due to hybridization. EMMOO produces the worst results due to not employing any meta-heuristics methods The proposed LEPSO algorithm strikes a balance by dynamically adjusting inertia weight and learning factors, thus improving the quality of results in terms of makespan, energy, and cost.

Table VI presents the average makespan, cost, and energy consumption for GAMPSO, EMMOO, PSO, GA, and LEPSO across all scenarios. The data shows that our proposed solution significantly enhances energy and makespan, with minimal cost enhancement.

As depicted in Fig. 4(a), When LEPSO is compared to GAMPSO, EMMOO, PSO, and GA in terms of the average Makespan, It achieves approximately 7%,74%, 32%, and 17% improvement respectively. Although all scenarios show gradual optimization, the worst-case scenario (scenario three) demonstrates the most substantial improvement in makespan due to the algorithm's use of non-linear inverse tangent particle acceleration and new learning factors.

Fig. 4(b) shows that the average cost is reduced by approximately 5% compared to PSO and by 1% compared to GA but 1% increased by GAMPSO. While cost reduction is observed in all scenarios except for scenarios three and four, the increase in communication costs due to increased task dependency among cloud and fog nodes may be further enhanced by implementing a new offloading method and adjusting weight constant assigned for cost specified in Table II. An intelligent offloading method may improve cost objectives as well.

Fig. 4(c) highlights that the LEPSO algorithm reduces energy consumption by about 74% compared to GAMPSO, about 98% compared to EMMOO, about 28% compared to PSO, and 84% compared to GA. This drastic reduction in energy consumption across all scenarios is achieved by assigning increased preference to the weight constant of cost, as specified in Table II. As a result, LEPSO selects those nodes that operate on minimum voltage supply and minimal frequencies, without compromising makespan or total cost. It is also evident that the meta-heuristics approach always achieves better in complex and multi-objective environments.

In addition to optimizing workflow scheduling, we also considered the algorithm's Running time as a crucial performance metric. running time, which refers to the CPU time allocated to the algorithm, was recorded for each algorithm across various scenarios, as shown in Table VII. Fig. 4(d) indicates that GA consumes significantly more CPU time as the workflow size increases, while Algorithms based on Particle Swarm Optimization utilized a more efficient use of CPU cycles. The GA's computational intensity and creation of unnecessary diversity in solutions lead to longer Running times. Although the LEPSO approach incurs slightly more running time compared to standard PSO due to the additional computation required for dynamically adjusting the learning factor and inverse tangent inertia weight in each iteration, this trade-off is justified by the significant reductions in makespan, cost, and energy.

D. Conversion of Evaluation Function

We observed and documented the fitness values for each particle update across all five scenarios. Fitness values were specifically monitored at every 10th generation of particle updates, as illustrated in Fig. 5. Two key observations emerged: first, although the population was initialized randomly for every method we compared, methods based on PSO consistently achieved minimal fitness values against the other methods. Second, the GA's fitness function did not converge even in

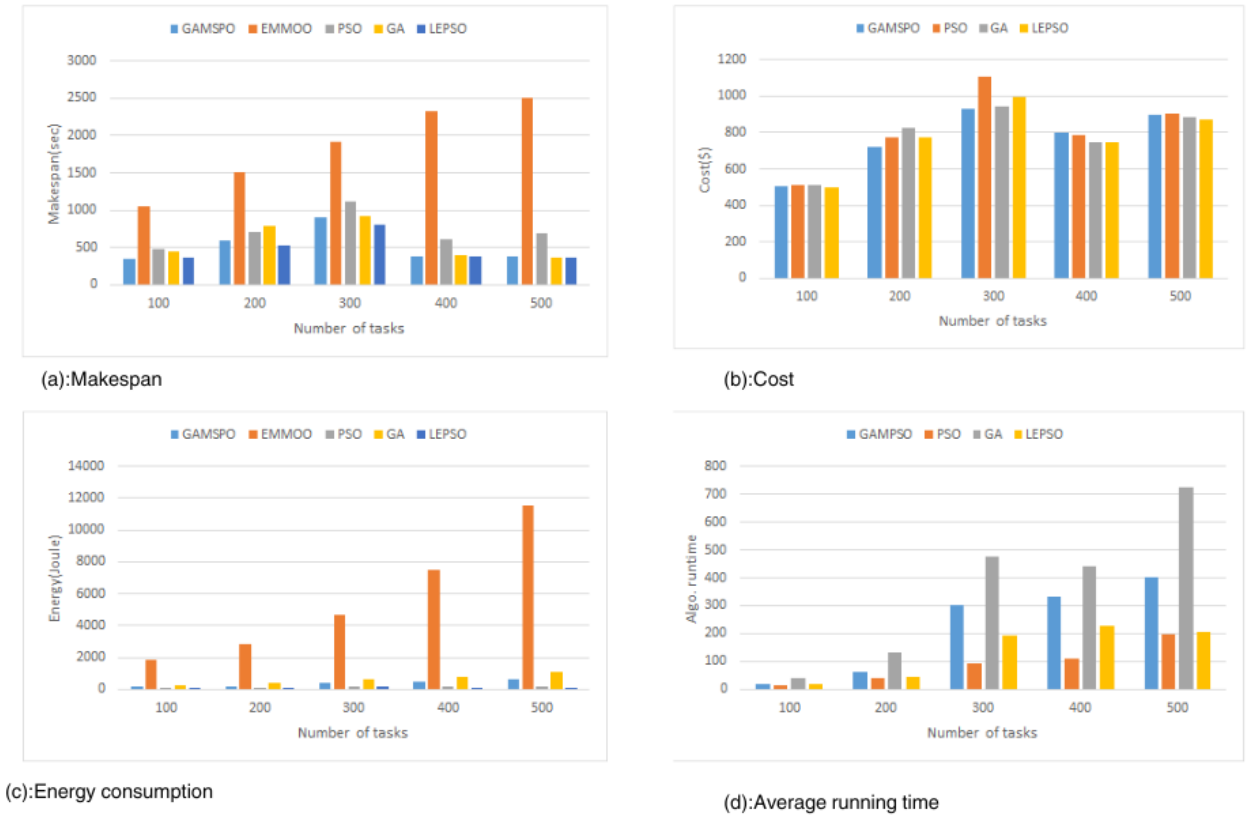


Fig. 4. Comparing algorithms on different objectives.

the final generations, resulting in suboptimal performance. In contrast, the standard PSO algorithm converged quickly but often got stuck in the local region, resulting to unreliable outcomes. Our proposed LEPSO algorithm strikes a good balance between reliability and effectiveness in the results.

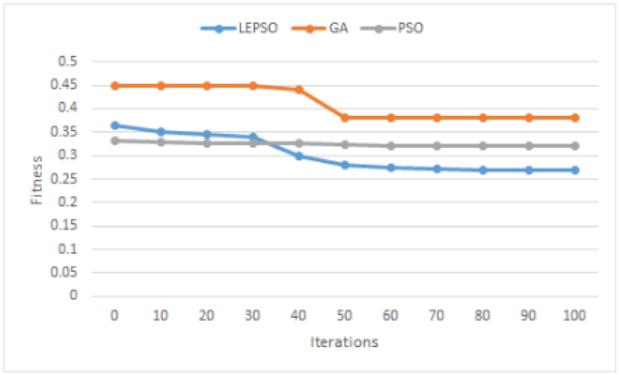
VII. Conclusions and Future Directions

The optimization of scheduling is a challenging problem, especially challenge becomes more complex when workflow applications are submitted cloud-fog environment to optimize objectives of a conflicting nature. PSO-based methods are a preferred choice because of their simplicity, rapid convergence, ease of implementation, and suitability for multi-objective optimization. However, PSO has some drawbacks, such as a tendency toward pre-mature convergence and becoming trapped in local regions, which can result in sub-optimal outcomes.

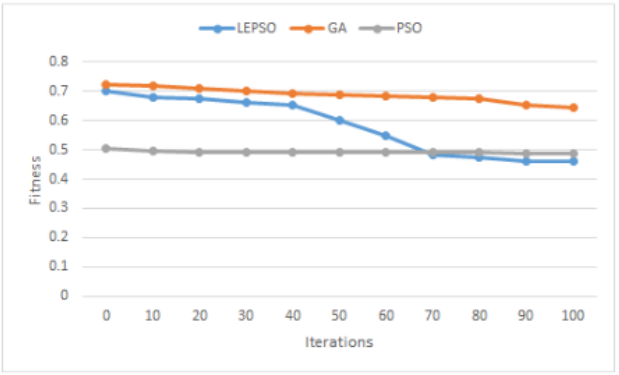
In this paper, we establish a theoretical framework to measure the performance metric of workflow applications that are to be scheduled to a fog-cloud system, addressing the conflicting nature of multi-objectives. We then introduce a Learning Enhanced Particle Swarm Optimization (LEPSO) algorithm for workflow scheduling, aiming to optimize conflicting objectives such as makespan, cost, and energy consumption. Two critical factors in PSO are inertia weight and learning factors significantly influence the particle's speed and velocity in the landscape. To address premature convergence and stagnation, the inertia weight of LEPSO is updated by applying an inverse tangent method and implementing a new learning factor method.

The effectiveness of LEPOS workflow scheduling is evaluated using Fogworkflow and compared its outcomes with the four recent algorithms: GAMPSO, EMMOO, GA, and PSO. The performance of LEPSO demonstrates superior in optimizing total cost, makespan, and energy consumption.

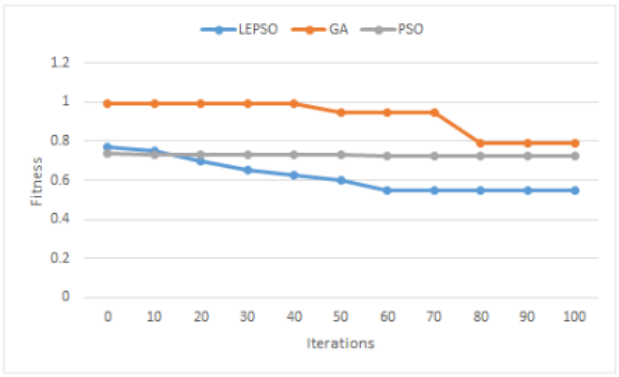
In future work, we plan to enhance this research by incorporating an intelligent offloading scheme and considering deadline constraints to better meet user requirements. Additionally, we intend to hybridize PSO with other meta-heuristic algorithms to improve its performance further. In this study, we have balanced multiple objectives using balance coefficients; however, we aim to implement a Pareto-optimal set in future



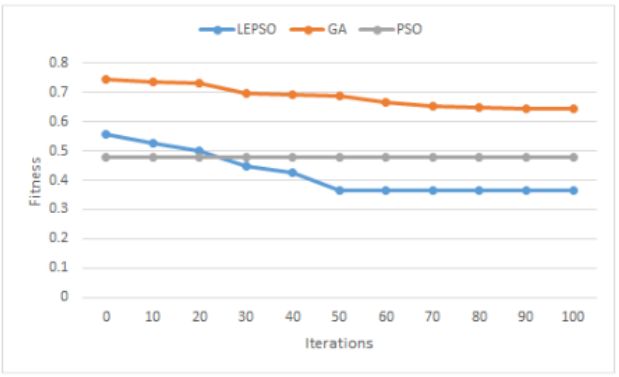
(a). for 100 tasks



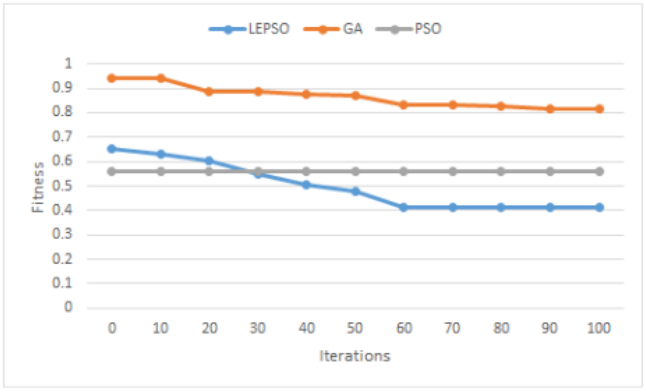
(b). for 200 tasks



(c). for 300 tasks



(d). for 400 tasks



(d). for 500 tasks

Fig. 5. Fitness optimization for [100-500] task.

work instead of relying on balance coefficients.

REFERENCES

[1] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, "Internet of things (iot): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities," *IEEE Internet of Things Journal*, vol. 8, no. 13, pp. 10 474–10 498, 2021.

[2] A. Čolaković and M. Hadžialić, "Internet of things (iot): A review of enabling technologies, challenges, and open research issues," *Computer networks*, vol. 144, pp. 17–39, 2018.

[3] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, 2019.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[5] N. Kumar, A. V. Vasilakos, and J. J. Rodrigues, "A multi-tenant cloud-based dc nano grid for self-sustained smart buildings in smart cities," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 14–21, 2017.

[6] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," pp. 13–16, 2012.

[7] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A survey," *ACM Transactions on Internet Technology (TOIT)*, vol. 19, no. 2, pp. 1–41, 2019.

[8] S. Kunal, A. Saha, and R. Amin, "An overview of cloud-fog computing: Architectures, applications with security challenges," *Security and Privacy*, vol. 2, no. 4, p. e72, 2019.

[9] H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," *Journal of Network and Computer Applications*, vol. 114, pp. 108–122, 2018.

[10] L. Belkhir and A. Elmeligi, "Assessing ict global emissions footprint: Trends to 2040 & recommendations," *Journal of cleaner production*, vol. 177, pp. 448–463, 2018.

[11] F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," *Future Generation Computer Systems*, vol. 78, pp. 257–271, 2018.

[12] G. Singh and A. K. Chaturvedi, "Particle swarm optimization-based approaches for cloud-based task and workflow scheduling: a systematic literature review," pp. 350–358, 2021.

[13] A. Mohammadzadeh, M. Masdari, and F. S. Gharehchopogh, "Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm," *Journal of Network and Systems Management*, vol. 29, no. 3, p. 31, 2021.

[14] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid ga-pso algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, pp. 1–16, 2018.

[15] M. Farid, R. Latip, M. Hussin, and N. A. W. Abdul Hamid, "A survey on qos requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, p. 551, 2020.

[16] 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," pp. 1114–1117, 2019.

[17] 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.

[18] S. Ijaz, E. U. Munir, S. G. Ahmad, M. M. Rafique, and O. F. Rana, "Energy-makespan optimization of workflow scheduling in fog–cloud computing," *Computing*, vol. 103, pp. 2033–2059, 2021.

[19] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft," *Future Generation Computer Systems*, vol. 93, pp. 278–289, 2019.

[20] K. K. Chakravarthi, L. Shyamala, and V. Vaidehi, "Cost-effective workflow scheduling approach on cloud under deadline constraint using firefly algorithm," *Applied Intelligence*, vol. 51, pp. 1629–1644, 2021.

[21] A. Iranmanesh and H. R. Naji, "Dchg-ts: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing," *Cluster Computing*, vol. 24, pp. 667–681, 2021.

[22] X. Xia, H. Qiu, X. Xu, and Y. Zhang, "Multi-objective workflow scheduling based on genetic algorithm in cloud environment," *Information Sciences*, vol. 606, pp. 38–59, 2022.

[23] M. Hosseinzadeh, M. Masdari, A. M. Rahmani, M. Mohammadi, A. H. M. Aldalwie, M. K. Majeed, and S. H. T. Karim, "Improved butterfly optimization algorithm for data placement and scheduling in edge computing environments," *Journal of Grid Computing*, vol. 19, pp. 1–27, 2021.

[24] N. Khaledian, K. Khamforoosh, S. Azizi, and V. Maihami, "Ikh-eft: An improved method of workflow scheduling using the krill herd algorithm in the fog-cloud environment," *Sustainable Computing: Informatics and Systems*, vol. 37, p. 100834, 2023.

[25] D. Javaheri, S. Gorgin, J.-A. Lee, and M. Masdari, "An improved discrete harris hawk optimization algorithm for efficient workflow scheduling in multi-fog computing," *Sustainable Computing: Informatics and Systems*, vol. 36, p. 100787, 2022.

[26] S. Nazir, S. Shafiq, Z. Iqbal, M. Zeeshan, S. Tariq, and N. Javaid, "Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid," in *Advances in intelligent networking and collaborative systems: the 10th international conference on intelligent networking and collaborative systems (INCoS-2018)*. Springer, 2019, pp. 34–46.

[27] A. Mohammadzadeh, M. Akbari Zarkesh, P. Haji Shahmohamd, J. Akhavan, and A. Chhabra, "Energy-aware workflow scheduling in fog computing using a hybrid chaotic algorithm," *The Journal of Supercomputing*, pp. 1–36, 2023.

[28] S. Bansal and H. Aggarwal, "A multiobjective optimization of task workflow scheduling using hybridization of pso and woa algorithms in cloud-fog computing," *Cluster Computing*, pp. 1–32, 2024.

[29] Y. Xie, Y. Zhu, Y. Wang, Y. Cheng, R. Xu, A. S. Sani, D. Yuan, and Y. Yang, "A novel directional and non-local-convergent particle swarm optimization based workflow scheduling in cloud–edge environment," *Future Generation Computer Systems*, vol. 97, pp. 361–378, 2019.

[30] L. Abualigah, A. Diabat, and M. A. Elaziz, "Intelligent workflow scheduling for big data applications in iot cloud computing environments," *Cluster Computing*, vol. 24, no. 4, pp. 2957–2976, 2021.

[31] R. Madhura, B. L. Elizabeth, and V. R. Uthariaraj, "An improved list-based task scheduling algorithm for fog computing environment," *Computing*, vol. 103, no. 7, pp. 1353–1389, 2021.

[32] M. I. Khaleel, "Hybrid cloud-fog computing workflow application placement: Joint consideration of reliability and time credibility," *Multimedia Tools and Applications*, vol. 82, no. 12, pp. 18 185–18 216, 2023.

[33] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 7, p. e5581, 2020.

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

[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," vol. 4, pp. 1942–1948, 1995.

[36] W. Chen and E. Deelman, "Workflowsim: A toolkit for simulating scientific workflows in distributed environments," pp. 1–8, 2012.

[37] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," pp. 1–10, 2008.

[38] J. C. Jacob, D. S. . Katz, T. Prince, B. G. Berriman, J. C. Good, A. C. Laity, E. Deelman, G. Singh, and M.-H. Su, *The Montage architecture for grid-enabled science processing of large, distributed datasets*. Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2004.