DOI: 10.24996/ijs.2026.67.1.39

ISSN: 0067-2904

Enhanced Manta Ray Foraging Algorithm for Scheduling Scientific Workflows in Cloud Computing Environments Using Levy Flight and Heuristic Operator

Mohanad Awad Abed*, Adnan Jumaa Jabir
Department of Computer Since, Collage of Since, University of Baghdad, Baghdad, Iraq

Received: 19/9/2024 Accepted: 28/1/2025 Published: 30/1/2026

Abstract

In modern computing, efficient task scheduling in cloud environments, especially for large-scale scientific workflows, presents a significant challenge as it is classified as a NP-hard problem. This study introduces an improved version of the Manta Ray Foraging Optimization Algorithm, named Lévy-Heuristic Manta Ray Foraging Optimization Algorithm (LH-MRFOA), which is enhanced with Lévy flight and heuristic search techniques to address these challenges. The Lévy flight mechanism is integrated to enhance the algorithm's exploration capabilities, allowing it to avoid local optima effectively and achieve global convergence. Meanwhile, the heuristic search method is employed to improve the exploitation capability of the algorithm while ensuring more efficient resource utilization and reduced processing time. The proposed LH-MRFOA, which mimics the natural foraging behavior of manta rays, combines these enhancements to deliver superior performance in task scheduling by minimizing makespan, processing cost, storage cost, and bandwidth utilization across varying workflow sizes. Experimental evaluations on a heterogeneous cloud infrastructure reveal that the LH-MRFOA outperforms bio-inspired algorithms such as Genetic Algorithm (GA) and Particle Swarm Optimization (PSO), particularly in scenarios that require high scalability and balanced resource allocation. This research substantially advances cloud task scheduling optimization, offering a robust solution for enhancing resource management and cost efficiency in real-world cloud applications.

Keywords: Cloud computing, Lévy flight, MET, MRFOA, Scientific workflows, Task scheduling, Workflow simulator.

خوارزمية شيطان البحر المحسّنة لجدولة سير العمل العلمي في بيئة الحوسبة السحابية باستخدام قفزة ليفي والعامل الاستدلالي

مهند عواد عبد*, عدنان جمعة جابر
قسم علوم الحاسوب، كلية العلوم، جامعة بغداد، بغداد، العراق

الخلاصة

في عالم الحوسبة الحديثة تُمثل جدولة المهام الفعّالة في البيئات السحابية واسعة النطاق تحديًا كبيرًا بحيث إنه تم تصنيفها على أنها مشكلة صعبة (متعددة الحدود). تقدم هذه الدراسة نسخة محسّنة من خوارزمية البحث عن الطعام لأسماك شيطان البحر(MRFOA) والتي تحاكي سلوك أسماك (Manta ray) اثناء بحثها

*Email: mohannad.abd2201m@sc.uobaghdad.edu.iq

عن الطعام ، والنسخة المحسنة تسمى خوارزمية (LH–MRFOA) المطورة بتقنيات البحث الاستدلالي لمعالجة هكذا نوع من التحديات حيث تم دمج آلية الاستكشاف لخوارزمية (Lévy flight) مع عامل الاستدلال (Heuristic search) لتحسين قدرات الاستكشاف لخوارزمية (MRFOA) مما يسمح لها بتجنب الوقوف عند الحلول المحلية بشكل فعال وتحقيق التقارب الأوسع وفي الوقت نفسهُ يتم استخدام طريقة البحث الاستدلالي لتحسين قدرة الاستغلال للخوارزمية مع ضمان استخدام الموارد بكفاءة أكبر وتقليل وقت المعالجة. تجمع خوارزمية LH–MRFOA المقترحة بين هذه التحسينات لتقديم أداء متفوق في جدولة المهام من خلال تقليل وقت التنفيذ، كلفة المعالجة وكلفة التخزين. وقد تم اختبار أداء الخوارزمية المحسنة خلال بيئات عمل مختلفة ولقد كشفت التقييمات التجريبية على البنية التحتية السحابية غير المتجانسة أن LH–MRFOA تتفوق على الخوارزميات المستوحاة من البيولوجيا مثل الخوارزمية الوراثية (GA) وخوارزمية أسراب الجسيمات (PSO)، وخاصة في السيناريوهات التي تتطلب قابلية عالية للتوسع وتخصيص الموارد بشكل متوازن. يوفر هذا البحث تقدمًا كبيرًا في تحسين جدولة المهام السحابية، مما يوفر حلاً قويًا لتعزيز إدارة الموارد وكفاءة التكلفة في تطبيقات السحابة في العالم الحقيقي..

1. Introduction

Presently, cloud computing serves as a key facilitator for digital transformation, changing how businesses consider IT services and infrastructures via the strategic importance of cloud computing in enhancing productivity, flexibility, and competitiveness in the digital era, especially with the present massive demand for cloud computing applications. The primary benefits of cloud computing are that users do not need to own or manage physical infrastructure, such as servers or networking equipment; instead, users can access computing resources on-demand from cloud service providers. This model offers several advantages, including cost savings, scalability, and resource allocation flexibility. Users can scale their resources up or down based on their needs, in addition to enforcing deadlines and constraints on tasks running within the cloud environment [1]. Cloud computing leverages virtualization technology to transform physical resources into virtual instances, streamlining the allocation and management of computing, storage, and networking resources. This abstraction enables seamless dynamic provisioning and scaling of resources in response to user demand, resulting in optimized resource utilization and cost-effectiveness.

Nevertheless, cloud computing encounters numerous obstacles, the most significant being the efficient utilization of computing resources. Efficient resource allocation is then translated into that the cloud provider can maximize the utilization of their infrastructure, as much as it enables users to obtain the best value for their investment in cloud services from performance to overall cost. Cloud orchestration systems, i.e., brokers and schedulers, map and assign workloads containing dependent and independent tasks to available resources in a process known as task scheduling. Task scheduling is critical in ensuring efficient resource utilization, optimal performance, and cost-effectiveness. It refers to the process of assigning incoming tasks to suitable virtual machines (VMs) or cloud resources within a cloud environment. Task scheduling complexity arises due to several factors, including the dynamic nature of cloud available resources, task dependencies such as scientific workflows, and user-defined constraints [2]. Such complexity makes reaching optimal task scheduling decisions regarded and recognized as NP-hard problem [3]. Effective task scheduling is crucial for achieving several key advantages in cloud computing i.e., enhancing performance, improving resource utilization, reducing costs, optimizing quality of services (QoS), and increasing scalability. As a result, finding an optimal scheduling solution typically requires exponential time in relation to the number of tasks and resources involved.

Scientific workflows are organized and represented using direct acyclic graphs (DAGs). In such representation, a task is defined as a node in the graph, and the dependencies among tasks (nodes) are represented as directed edges.

Scheduling optimization of scientific workflows has received great attention within academic literature, and it is an active and evolving field of research that emphasizes the importance of efficient cloud resource utilization [4][5]. Such interest is primarily driven by the widespread adoption of cloud-based services and solutions and the important role scheduling optimization plays in cost minimization and efficiency of cloud resource utilization. To address this challenge, researchers explored heuristic, metaheuristic, and several nature-inspired optimization algorithms to approximate near-optimal task scheduling solutions within reasonable time and other user defined constraints. Upon examining recent research [9-12], it is evident that various improvements have been made to enhance the convergence of several optimization algorithms to determine the optimal global-best task scheduling solution within cloud environments.

However, several limitations and drawbacks have been identified. Primarily, the research often confines its evaluation to small-scale test datasets, limiting the generalization of the findings. Second, several of the proposed hybrid approaches frequently exhibit high computational complexity, while others achieved marginal improvements compared to alternative optimization algorithms. The cumulative impact of these limitations can be significant in convergence to optimal solution in the optimization of large-scale tasks. Last, a certain number of studies suffer from poor selection of virtual machines configuration to accommodate the aim of their study, such as employing a small number of virtual machines in large and high-performance tasks scheduling cloud environments or an excessive number of virtual machines to investigate scheduling of small and limited number of tasks. From this standpoint, the main contributions of our work comprise:

1. To enhance the efficiency of scientific workflow task scheduling in cloud environments, emphasizing minimizing both makespan and costs (including processing, bandwidth, and storage) across workflows of various sizes (small, medium, and large).
2. The Manta Ray Foraging Optimization Algorithm (MRFOA) is introduced as a solution for optimizing task scheduling. MRFOA is recognized for its robust and fast convergence in large-scale optimization problems. Thus, MRFOA is particularly well-suited for handling the complexities of large-scale multi-objective task scheduling in cloud environments
3. An enhanced version of the Manta Ray algorithm, termed Lévy-Heuristic Manta Ray Optimization Algorithm (LH-MRFOA), is proposed. LH-MRFOA incorporates Lévy flight randomness and heuristic approach to enhance further the convergence towards global near to optimal solution, hence, more suited to address optimization challenges inherent in complex workflows.
4. The proposed algorithm undergoes comprehensive tests on heterogeneous cloud infrastructure characterized by different processing speeds (slow, moderate, fast) and costs. This rigorous examination aims to thoroughly evaluate the effectiveness of the proposed approach in real-world cloud computing scenarios.

The subsequent sections of this paper are organized as follows: Section 2 thoroughly examines recent literature on multi-objectives task scheduling optimization with an analysis of various methodologies employed and critical evaluations of their weaknesses. Section 3 details the definition of the problem and the objective function in scientific workflow task scheduling. Section 4 presents MRFOA as the proposed algorithm and proposed improvements by employing Lévy flight-heuristic as a search factor. Our experimental setup is explained in Section 5. The evaluation of performance is discussed in Section 6, while Section 7 encompasses the conclusion and outlines avenues for future research endeavors.

2. Related Work

Scheduling in cloud computing has been an object of extensive research since it has been classified as an NP-hard problem. Such complexity lies in the combinatorial nature inherent

to the resource allocation and coordination of task execution. Researchers have been considering different heuristic and metaheuristic approaches for solutions [6], with a special emphasis on optimization techniques recently. Y. Zhang et al., 2023 [7] proposed a Dynamic Multi-Objective Evolutionary Algorithm (DMOEA) for workflow scheduling in dynamic cloud environments. The work has emphasized the necessity of adaptation to the dynamically priced spot resources and was focused on the maximization of reliability and minimization of cost and makespan. However, the approach has high computational and space complexity, which is unsuitable for large-scale datasets. Adnan Talha et al. (2022) [8] proposed Oppositional-Based Learning (OBL) integrated with the Pathfinder Algorithm (PFA) to improve the performance of task scheduling in large-scale workflows. Although the hybrid approach improved the exploration and exploitation abilities, the study did not have theoretical advances in the chosen algorithms.

N. Manikandan et al. (2022) [9] carried out task scheduling using a hybrid Whale Optimization Algorithm (WOA) combined with mutation-based Bees optimization. Despite achieving remarkable improvement in resource utilization and operational cost, the method showed high computational time for large-scale scenarios, but it was not tested on benchmark datasets. Haithem Hafsi et al. [10], in 2022, proposed the genetically modified multi-objective particle swarm optimization algorithm with a novel two-dimensional encoding for task-resource mapping in high-performance hybrid cloud environments. While the approach has reached faster solution convergence, it was only tested with a small number of virtual machines and medium-scale datasets, which can hardly be generalized.
Junlong Zhou et al. (2019) [11] proposed an improved genetic algorithm for hybrid cloud scheduling to minimize cost and makespan under SLA constraints. The two-stage approach in this work improved solution quality at the expense of heavy computational overhead and, hence, was unsuitable for a dynamic environment with large-scale workflows. To deal with large-scale optimization, Bilal H. Abed-Alguni et al. (2021) [12] introduced a Distributed Grey Wolf Optimizer (DGWO). The technique organized candidate solutions into islands for parallel evaluation, improving the exploration capabilities. However, the approach lacked statistical analysis of migration strategies, which reduced the insight into solution quality for high data transmission tasks.

J. Kok Konjaang et al. (2021) [13] proposed a three-stage task scheduling method combining the Cost Optimized Heuristic Algorithm (COHA) and Multi-Objective Workflow Optimization Strategy (MOWOS). Despite achieving cost and makespan reductions, the study did not include bandwidth and storage metrics in its evaluation, making it less comprehensive. In QoS-oriented optimization, Xianyong Wei (2020) [14] proposed an ACA with dynamic pheromone update strategy and load balancing strategies. While the method achieved a very high improvement in dynamic resource allocation, it still suffered from small-scale task evaluation and homogeneous virtual machine configurations. Hatem Aziza et al. [15], 2020, integrated the Heterogeneous Earliest Finish Time (HEFT) with the Genetic Algorithm for scientific workflows and proposed new crossover and mutation operators. However, the fitness function was very basic, and the approach did not significantly improve in most of the test cases. Ali Mohammad Zadeh et al., 2021 [16] used the Sine Cosine Algorithm with chaotic randomness to modify the Ant Lion Optimization algorithm to green cloud computing. While the approach was efficient at a low workload, computational overhead was high at a high workload.

Table 1: A Comparison of the Current Workflow Task Scheduling

Ref.	Metrics	Method	Limitations
[7]	Reliability, makespan, cost	Dynamic multi-objective optimization evolutionary algorithm (DMOEA).	. High computational complexity . High space complexity (weights vectors) . Only small and medium size datasets were tested.
[8]	Makespan, cost, resource utilization	Hybrid Pathfinder and oppositional-based Learning (OBLPFA).	. No theoretical improvements in the selected algorithms.
[9]	Makespan, cost, energy consumption	Hybrid Whale and mutation-based Bee optimization algorithms.	. High computational and time complexity due to double adaptive weight and random spare scheme, especially for large scale workflow tasks. . Workflow standard benchmark datasets were not tested.
[10]	Makespan, cost, SLA factors such as budget	Genetically modified Particle swarm algorithm (GMPSO) and novel two-dimensional encoding for task and resource mapping.	. The proposed work aimed at high performance hybrid cloud scenarios, yet only six VMs were configured (one free and five paid). A large number of VMs must be tested. . Synthetic extra-large workflow tasks should be considered such as [8] and [14]. . In terms of evaluation metrics IGD (inverted generational distance) and Hv (Hypervolume), the performance of the proposed GMPSO is relatively close, if not worse, when compared to NSGAII. Better results are obtained only in higher iterations.
[11]	Makespan, monetary cost, SLA constraints.	Enhanced genetic algorithm (improved chromosome encoding and hybrid crossover).	. The proposed two-stage solution is slow in a dynamic cloud environment. . Only small workflow workloads were tested.
[12]	Computation and data transmission costs.	Distributed grey wolf optimizer (DGWO).	. The proposed distribution is to increase the number of parallel workflow evaluations and not to enhance the GWO algorithm. . The maximum tested data transmission is 5GB, a bigger value should be considered. . No statistical analysis of solutions migration among islands. . The impact of the best and worst solutions from one island to another was not presented.
[13]	Makespan, cost	Hybrid cost optimized heuristic and multi-objective workflow optimization strategy (MOWOS), with improved task schedular using MinVM and MaxVM.	. Tasks splitting approach effect was not measured during the evaluation. It must be included in the Fitness function. . Tasks splitting was based on task length, which is accommodated by splitting the task's bandwidth, storage, each of which was not covered. . MinVM and MaxVM scheduling effects on the overall obtained results were not measured.
[14]	Makespan, cost, energy, SLA deadline constraints	Hybrid Chemical Reaction Optimization (CRO), and Particle Swarm Optimization (PSO).	. Proposed Cloudlets properties share the same file input and outsize. . Presented VMs' configurations are not heterogeneous and are relatively the same. . The maximum number of tested tasks is 300, large tasks number should be tested. . No standard workflow benchmark datasets were tested. . The time complexity of the proposed CR-PSO is high
[15]	Dynamic resources availability, dynamic prices of cloud	Improved Ant Colony Algorithm (ACA) with reward and punishment coefficient to enhance pheromone updating strategy	. A large number of tasks should be considered, the maximum number of tasks tested was only 200. . The number of VMs in the experimental results was 80, which is rather large considering the small number of tasks tested.

	resources, makespan, QoS	of ant colony.	. No standard workflow benchmark datasets were tested.
[16]	Makespan, cost, deadline budget	Hybrid Genetic Algorithm and Heterogeneous earliest finish time HEFT, tournament crossover, and random mutation operator.	. The proposed approach can be improved, and better results were obtained in only half of the performed workflow tests. . The proposed Fitness function is composed of two subfunctions and it can be improved
[17]	Energy, makespan, and resources cost	Improved Ant Lion Optimization (ALO) using Sine Cosine Algorithm and Chaotic randomness.	. The number of VMs' in the experimental setup is set to 1000 VM. . The largest workflow tested load only contain 1000 tasks, bigger workflow should be considered given the large number of VMs, such as [5] and [10].

3. Scientific Workflow Task Scheduling

Scientific workflows are sequences of tasks used by scientific research to test the efficiency of task scheduling algorithms in the cloud environment. Workflows are systematic sequences of activities or tasks aimed at generating scientific results or solving complex problems. Workflows are typically represented in a directed acyclic graph (DAG). The term "acyclic" implies no cycles or loops in the graph, meaning that tasks can be executed in a specific order without encountering circular dependencies. Each node in the graph represents a task or a computation along with other task related attributes, such as task length, and data required for input and output. Edges, on the other hand, indicate the dependencies between tasks. An edge from task A to task B indicates that task B depends on the output of task A. This dependency structure ensures that tasks are executed in the correct order, with prerequisite tasks completed before their dependent tasks can begin. Fig 1 [17] shows the DAG representation of five scientific workflows utilized in this study: Montage (a), CyberShake (b), Epigenomics (c), Inspiral (d), SIPHT (e).

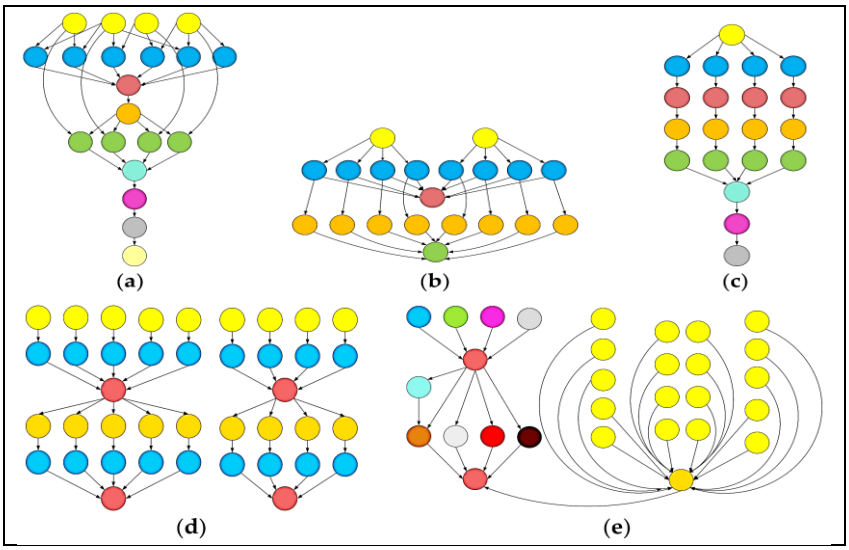


Figure 1: DAG structure of scientific workflows [17].

DAG is mathematically represented as $G = (T, E)$. Where $T = \{T_1, T_1, T_2, \ldots\ldots, T_n\}$ denote the set of vertices or nodes (tasks). And E represents the set of directed edges among nodes, where $E = \{E_0, E_1, E_2, \ldots\ldots, E_n\}$. For instance, a directed edge of $E(T_3, T_4)$ indicates a direct dependency between task T_3 and T_4. In other words, the fourth task cannot be executed until the third task has been completed. Therefore, E can be rewritten as a set of ordered pairs of vertices, $E = \{ (T_i, T_j) \mid T_i, T_j \in T\}$. Where T_i is the predecessor task and T_j is successor task.

Each task T_i in DAG workload has additional attributes or metadata. These attributes help in optimizing task execution and resource allocation within the workflow, such as:

- **Input data:** $I = \{I_0, I_1, I_2, \dots\dots I_n\}$, where I_i represents input data required by task T_i.
- **Output data:** $O = \{O_0, O_1, O_2, \dots\dots O_n\}$, where O_i represent output data produced by task T_i.
- **Task length:** $length(T_i)$, is a measure of computational effort required to complete task T_i. It is typically expressed in terms of the number of instructions or the amount of competition (e.g., in millions of instructions). In workflow simulations, task length is used to estimate the makespan or the time required to complete the task, assuming certain computational resources are available.

In the realm of Infrastructure as a Service (IaaS), computing resources are commonly provisioned in the form of virtual machines (VMs). These VMs are simulated computing environments operating independently within a physical server. Users of IaaS platforms leverage these VMs to deploy and run their applications, software, and computational workloads in a flexible and scalable manner.

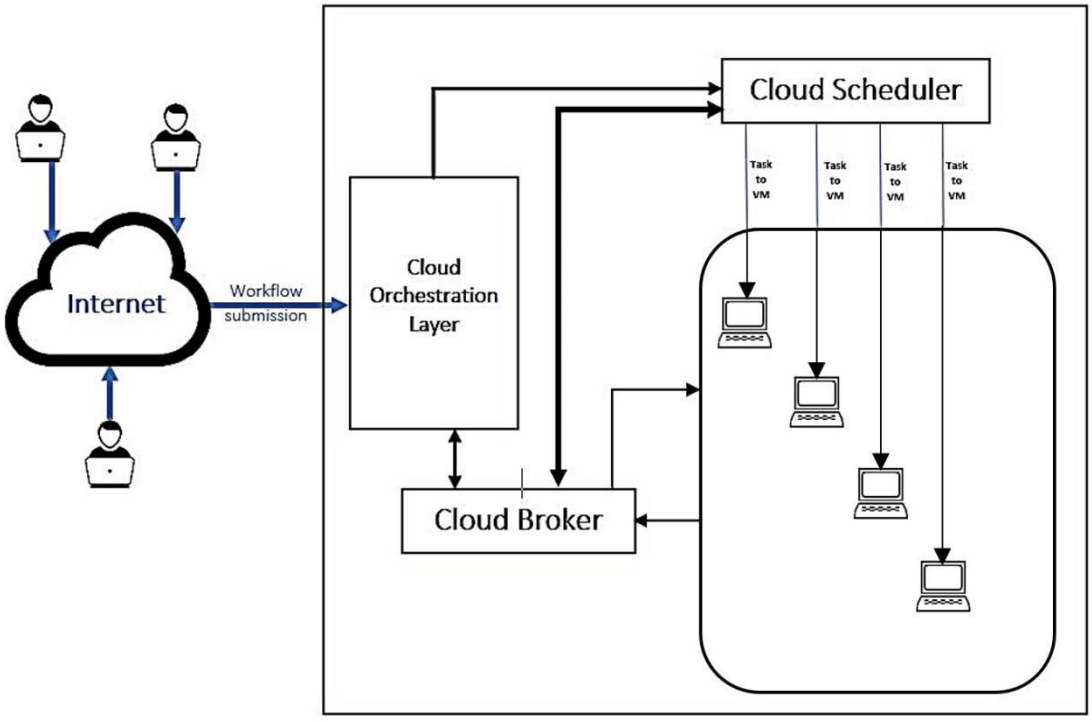


Figure 2: Workflow and Cloud tasks scheduling

3.1 Problem Formulation

The context of this study aims to present and develop a multi-objective task scheduling optimization algorithm. This algorithm aims to address the complexities inherited in task scheduling within cloud computing environments. By the above given definitions for the workflows and tasks, our primary objective is to minimize critical factors associated with task scheduling on cloud resources. These factors encompass the makespan, processing costs, storage costs, and bandwidth utilization. Each of these factors can be defined as follows:

1. Makespan: the total time required to complete all tasks within a given workflow, starting from the initiation of the first task to the completion of the last task. Consider the following workflow, $G = (T, E)$, where $T = \{T_1, T_1, T_2, \dots, T_n\}$ represent the set of tasks (both

dependent and independent), and E is a set of all edges or dependency connections among tasks. Each connection is denoted as $E_i = (T_i, T_j)$, and belongs to $E = \{E_0, E_1, E_2, \dots., E_n\}$.
A task $T_i$with dependencies, dep (T_i) will not start unless all its dependent tasks have been executed. If a task has no dependencies (e.g., entry tasks), dep $(T_i) = \emptyset$. The finish time FT_{T_i} for task T_i is defined as the sum of its execution time and the execution times of all dependent tasks [12]:

$$FT_{Ti} = (\sum_{i=0}^{n} ET\ (dep(T_i)) + \ ET(T_i) \quad (1)$$

Where n is the number of dependent tasks and $ET = length(T_i)/(MPIS_{vmj} * PE_j)$. Where $MIPSvm_j$ is processing power of VM_j measured in $MIPS$, and PE_j is the number of available virtual cores (*Processing Entity)*; Makespan can be measured as:

$$Makespan = max_{\forall tasks}(FT) \quad (2)$$

Where $FT = \{FT_0, FT_1, FT_2, \dots, FT_n\}$.

2. Processing Cost: The cost associated with executing task T_i on resource VM_j. In workflow simulation, processing cost P_i for task T_i is calculated as [12]:

$$P_i = ET_i * Cost_j \quad (3)$$

Where ET_i is execution time for task T_i and $Cost_j$ is the *Processing Cost* of VM_j per time unit. The total processing cost for the entire workload can be determined as:

$$PCost_{total} = \sum_{i=0}^{n} P_i \quad (4)$$

3. Storage Cost: The expenditure associated with storing data related to tasks on cloud storage services. The storage requirement S_i for the task T_i, is the sum of all its output file sizes hence [12]:

$$S_i = \sum_{j=0}^{n} O_{ij}$$

Where O_{ij} is the j_{th} output file for the task T_i. Then *storage cost* SC_i for task T_i is:

$$SC_i = (\ S_i\ /\ SVM_j)\ * SCVM_j$$

Where SVM_j is the total storage available for VM_j, and $SCVM_j$ is the *storage cost* of VM_j.
the total storage cost for the workflow is:

$$Total\ storage\ cost = \sum_{i=0}^{n} SC_i$$

4. Bandwidth Cost: Refers to the network bandwidth consumption during data transfer operations in the cloud. The required bandwidth B_i for the task T_i is the sum of all input file sizes [12]:

$$B_i = \sum_{j=0}^{n} I_{ij}$$

Where I_{ij} is the j_{th} input file for the task T_i. Then *bandwidth cost* BC_i for task T_i is:

$$BC_i = (\ B_i\ /\ BVM_j)\ * BCVM_j$$

Where BVM_j is the total bandwidth available for VM_j, and $BCVM_j$ is the *bandwidth cost* of VM_j. The total bandwidth cost for the workflow is:

$$Total\ bandwidith\ cost = \sum_{i=0}^{n} BC_i$$

3.2 Objective Function

In this research, we aim to decrease task execution time and cost while maximizing resource utilization across all VMs in a cloud environment. Workflow scheduling in dynamic environments is more challenging and realistic compared to static environments. Cloud resources must be allocated precisely to fulfill user service agreements while maximizing resource utilization. Large scale with multi-objectives optimization is inherently complex, and optimal solutions require efficient scheduling algorithms and more importantly, accurate objective function. Considering this, the objective function of this study is focused primarily on minimizing the following fitness function:

$$F = (w_1 * MS + w_2 * PC_{total} + w_3 * SC_{total} + w_4 * BC_{total})$$

Where MS is $MakeSpan$, PC_{total} is the total *Processing Cost*, SC_{total} is the total *Storage Cost*, and BC_{total} is the total *Bandwidth Cost*. And w_i represents the weight, $w_i \in [0, 1]$, and $w_1 + w_2 + w_3 + w_4 = 1$.

Weights are used in optimization functions to assign relative importance or priority to different objectives or criteria. In a cloud environment, such decisions are entirely dependent on user preferences and on the specification of the workflow at hand. Objectives with higher weights are given more emphasis, leading to solutions prioritizing those objectives over others. Additionally, it provides against contradictory objectives such as makespan and processing costs.

4. Proposed LH-MRFOA Algorithm

This section presents the original MRFO algorithm along with the proposed improvements.

4.1 Manta Ray Foraging Optimization Algorithm (MRFOA)

Manta Ray Foraging Optimization Algorithm (MRFOA) is a metaheuristic optimization method first proposed by Zhao in 2020. It is inspired by the effective and cooperative foraging behavior of manta rays. MRFOA has successfully imitated three major foraging behaviors of manta rays, including chain foraging, cyclone foraging, and somersault foraging. The behavior is customized for optimization tasks with a good balance between exploration and exploitation to find the global optimum in complex search spaces.

4.1.1 Manta Ray Structure and Foraging Behavior

Manta rays are large, flat, aquatic animals with terminal mouths; they forage using their cephalic lobes to direct the plankton into the mouth. Manta rays portray peculiar foraging behavior, such as looping and somersaulting motions. Figures 3(A) and 3(B) [20] depict a manta ray in action and its structure.

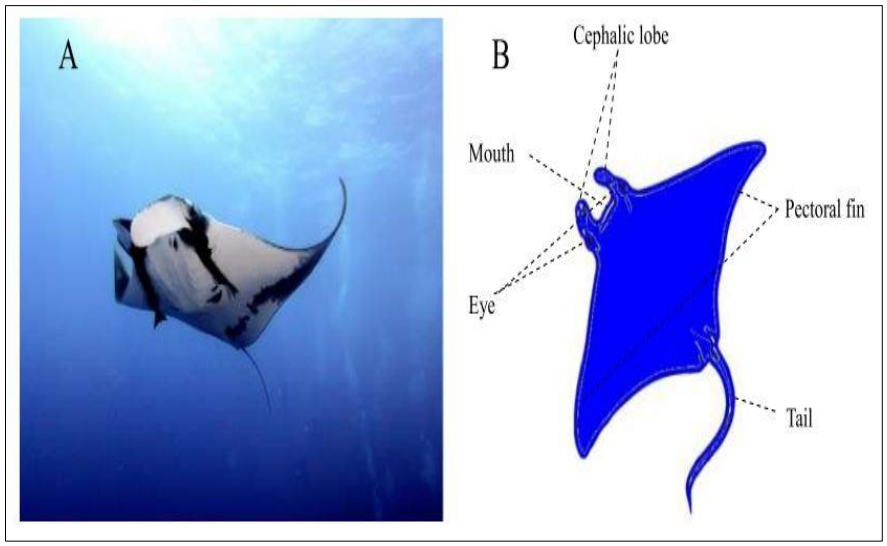


Figure. 3: (A) A foraging manta ray, and (B) structure of a manta ray [20]

Chain foraging, Figure 4 depicts the 2-D foraging behavior of cyclones. Instead of merely following the meal in front of it, the individual moves in a circular pattern in that direction. In two dimensions, the spiral motion of manta rays is described by the mathematical expression:$x_i^d(t+1) =$

$$\begin{cases} x_i^d(t) + r.\left(x_{best}^d(t) - x_i^d(t)\right) + \alpha\,.\left(x_{best}^d(t) - x_i^d(t)\right) & i = 1 \\ x_i^d(t) + r.\left(x_{i-1}^d(t) - x_i^d(t)\right) + \alpha\,.\left(x_{best}^d(t) - x_i^d(t)\right) & i = 2,\dots,N \end{cases} \quad (1)$$

$$\alpha = 2\,.r.\sqrt{|\log(r)|} \quad (2)$$

where, $x_i^d(t)$ is the position of ith individual at time t in dth dimension, r is a random vector within the range of [0,1], a is a weight coefficient, $x_{best}^d(t)$ is the plankton with high concentration. Fig. 4 depicts this foraging behavior in a 2-D space. The position update of the ith individual is determined by the position $x_{i-1}(t)$ of the (i-1) th current individual and the position $x_{best}(t)$ of the food.

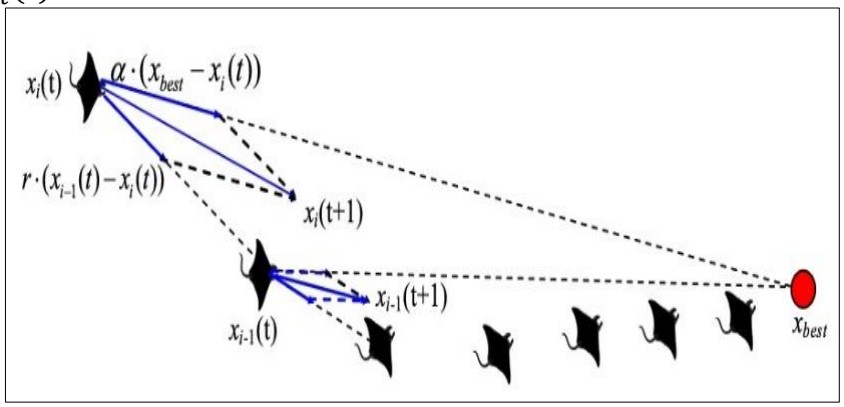


Figure 4: Chain foraging behavior in 2-D space [20]

- **Cyclone foraging**: Figure 4 depicts the 2-D foraging behavior of cyclones. Instead of merely following the meal in front of it, the individual moves in a circular pattern in that direction. In two dimensions, the spiral motion of manta rays is described by the mathematical expression:

$$\begin{cases} x_i(t+1) = x_{best} + r.\left(x_{i-1}(t) - x_i(t)\right) + e^{bw}\,.\,cos(2\pi w)\,.\,x_{best} - x_i(t)\right) \\ x_i(t+1) = x_{best} + r.\left(x_{i-1}(t) - x_i(t)\right) + e^{bw}\,.\,sin(2\pi w)\,.\,x_{best} - x_i(t)\right) \end{cases}$$

(3)

where w is a random number in [0, 1], this motion behavior may be extended to a n-D space. For simplicity, this mathematical model of cyclone foraging can be defined as:

$$x_d^i(t+1) = \begin{cases} x_{best}^d + r.\left(x_{best}^d(t) - x_i^d(t)\right) + \beta.\left(x_{best}^d(t) - x_i^d(t)\right) & i = 1 \\ x_{best}^d + r.\left(x_{i-1}^d(t) - x_i^d(t)\right) + \beta.\left(x_{best}^d(t) - x_i^d(t)\right) = & i = 2, \dots, N \end{cases} \quad (4)$$

$$\beta = 2e^{r1\frac{T-t+1}{T}}.\sin(2\pi r_1) \quad (5)$$

where r_1 is the rand number in [0, 1], T is the maximum number of repetitions, and β is the weight coefficient.

The equation for this mechanism may be seen below. As seen in the illustration below, it focuses mostly on exploration, enabling MRFO to conduct an extensive worldwide search:

$x_{rand}^d = Lb^d + r.(Ub^d - Lb^d)$ (6)

$$x_d^i(t+1) = \begin{cases} x_{best}^d + r.\left(x_{best}^d - x_i^d(t)\right) + \beta.\left(x_{rand}^d - x_i^d(t)\right) & i = 1 \\ x_{best}^d + r.\left(x_{i-1}^d - x_i^d(t)\right) + \beta.\left(x_{rand}^d - x_i^d(t)\right) & i = 2, \dots, N \end{cases} \quad (7)$$

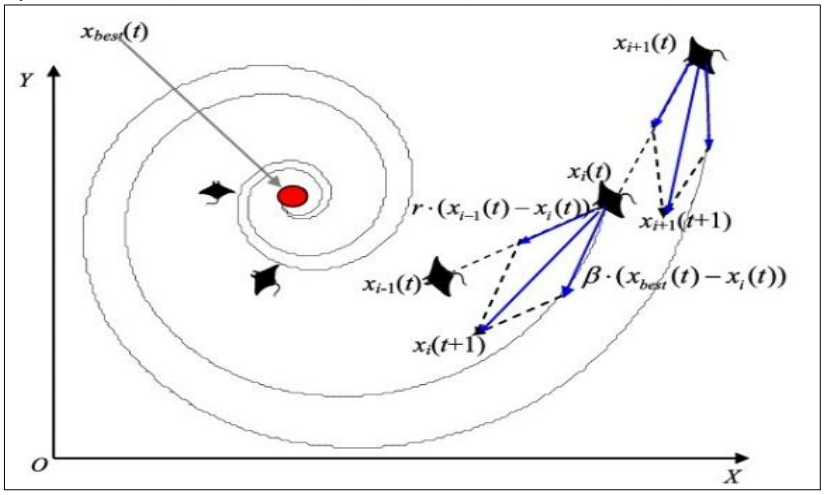


Figure.5: Cyclone forging behavior in 2-D space [20]

- **Somersault foraging*:*** The food's location is seen as a pivot in this behavior. Every manta ray tends to swim around the pivot and somersault into a different position. As a result, they constantly adjust their positions to reflect the best position thus far. The following is one way to develop the mathematical model:

- $x_i^d(t+1) = x_i^d(t) + S.\left(r_2.x_{best}^d - r_3.x_i^d(t)\right),\ i = 1, \dots, N$ (8)

where *S* is the somersault factor that decides the somersault range of manta rays and *S* = 2, r_2 and r_3 are two random numbers in [0, 1].

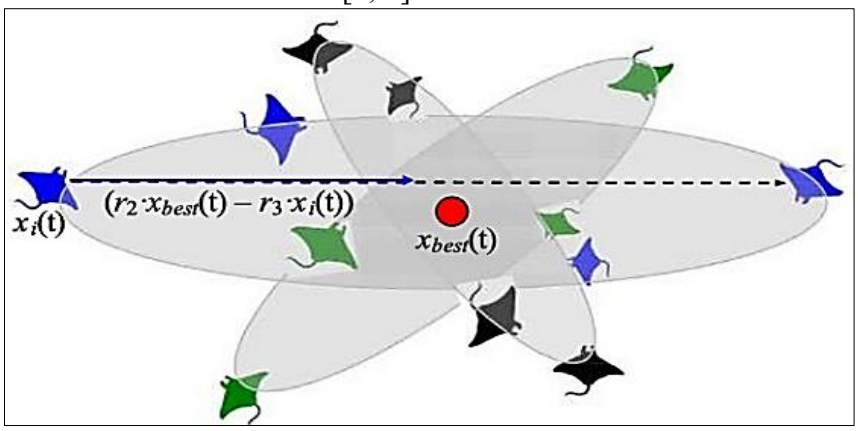


Figure 6: Cyclone foraging behavior in 2-D space [20]

Algorithm I, shows the pseudo code for the basic specifications of the MRFO algorithm [20]

Algorithm I: MRFO Algorithm

Input Parameters//

N: Population size (number of manta rays).
T_max: Maximum number of iterations.
x_l: Lower boundary of the search space (problem domain).
u_x: Upper boundary of the search space (problem domain).
S: Somersault factor.
Fitness Function f(x): Objective function to be minimized or maximized.

Output//

x_best : The best solution found by the algorithm.
f(x_best): The fitness value of the best solution.

Initialize population //

Initialize the size of population *N*, the maximal number of iterations *T* and each manta ray $x_i(t) = x_l + rand\ (x_u - x_l)$ for i=1 …., *N* and t=1.

Compute the fitness of each individual $f_i = f(x_i)$ and obtain the best solution found so far $x_{best,}$ Where $x_u\ and\ x_l$ are the upper and lower boundaries of problem space, respectively.

Iterative optimization //

WHILE stop criterion is not satisfied do

FOR i=1 TO *N* DO

IF rand <0.5 THEN //*Cyclone foraging*

IF $t/T_{max} < rand$ THEN

$x_{rand} = x_l + rand\ .(x_u - x_l)$

$$x_i(t+1) = \begin{cases} x_{rand} + r.\left(x_{rand} - x_i\ (t)\right) + \beta\ .\left(x_{rand} - x_i\ (t)\right) & i = 1 \\ x_{rand} + r.\left(x_{i-1}\ (t) - x_i\ (t)\right) + \beta\ .\left(x_{rand} - x_i\ (t)\right) & i = 2, \dots, N \end{cases}$$

ELSE

$$x_i(t+1) = \begin{cases} x_{rand} + r.\left(x_{best} - x_i\ (t)\right) + \beta\ .\left(x_{best} - x_i\ (t)\right) & i = 1 \\ x_{rand} + r.\left(x_{i-1}\ (t) - x_i\ (t)\right) + \beta\ .\left(x_{best} - x_i\ (t)\right) & i = 2, \dots, N \end{cases}$$

END IF.

ELSE // *Chain foraging*

$$x_i(t+1) = \begin{cases} x_{rand} + r.\left(x_{best} - x_i\ (t)\right) + \alpha\ .\left(x_{best} - x_i\ (t)\right) & i = 1 \\ x_{rand} + r.\left(x_{i-1}\ (t) - x_i\ (t)\right) + \alpha\ .\left(x_{best} - x_i\ (t)\right) & i = 2, \dots, N \end{cases}$$

END IF.

Compute the fitness of each individual $f(x_{i(t+1)})$

IF $f(x_{i(t+1)}) < f(x_{best})$

THEN $x_{best} = x_i(t+1)$

END IF

// *Somersault foraging*

FOR i=1 TO *N* DO

$x_{i(t+1)} = x_{i(t)} + S\ .\left(r_2.x_{best} - r_3\ .x_{i(t)}\right)$

Compute the fitness of each individual $f(x_{i(t+1)})$

IF $f(x_{i(t+1)}) < f(x_{best})$

THEN $x_{best} = x_i(t+1)$

END IF

END FOR.

END WHILE.
Return the best solution found so far x_{best}

In several technical disciplines i.e., geophysics [21], energy allocation [22], image processing [23], and electric power [24], the MRFO algorithm performs exceptionally well and exhibits a wide range of optimization talents. The MRFO has proven to be a conducive approach to resolving several intricate real-world issues through these successful applications [25]. Despite being a member of the meta-heuristic algorithm domain, MRFO differs greatly from other popular meta heuristics in philosophy and design. i.e., in comparison with Particle Swarm Optimization (PSO), MRFO and PSO they differ primarily in their biological actions. While MRFO draws inspiration from manta ray social foraging activities, PSO is motivated by the movement of bird flocks in the natural world. The way the two hunt for solutions differs significantly from one another [26]. The global best solution is combined with other solutions to create PSO solutions. A further notable distinction between the two is the method used to look for solutions. The global best solution found so far, the local best solution, and the individual movement velocities combine to produce the solutions in PSO; in contrast, the global best solution found so far, and the solution in front of it combine to produce the solutions in MRFO by switching different movement strategies [27]. Another comparison in the same domain with Genetic Algorithm (GA) in contrast to the communal foraging activities of manta rays in MRFO, GA is based on Darwin's theory of evolution. The representation of problem variables is the second distinction. Whereas the issue variables in MRFO are utilized directly, in GAs, they are represented as a sequence of fixed-length bit strings. Better solutions also have a higher chance of generating new solutions in GAs when the roulette wheel selection approach is used, and inferior solutions will likely be replaced by better new solutions [28]. While in MRFOA every member of the population has an equal chance of improving their solutions.

4.2 Improved MFROA

Our proposal for a hybrid Lévy-Heuristic Manta Ray Foraging Optimization Algorithm for multiple critical objectives in workflow scheduling: makespan minimization, processing cost reduction, storage cost optimization, and bandwidth utilization minimization. One of the new improvements embeds the Lévy flight mechanism in the MRFOA, enabling it to take big probabilistic leaps in search space. This addition elevates the exploration capabilities greatly and therefore escapes local optimum and finds globally optimal solutions. On the other hand, we suggest a different enhancement that involves addressing dependency management. This process begins by iterating over each task and identifying whether it has any dependent tasks (parent tasks). If a task has no parent tasks (independent tasks), it is processed directly without needing dependency handling. However, if a task has one or more parent tasks (dependent tasks), the algorithm will initiate a procedure to optimize resource allocation for these dependent tasks. The Heuristic Dependency Management (HDM) mechanism is the second enhancement in LH-MRFOA, addressing task dependencies in workflow scheduling. It focuses on optimizing the allocation of dependent tasks to ensure efficient resource utilization and adherence to dependency constraints. Initially, the dependent tasks are temporarily stored and cleared from previous iterations to ensure accurate processing without conflicts. For each parent task, the algorithm evaluates the performance of assigning the task to different virtual machines (VMs). It calculates the expected execution time for each task on every available VM based on the task's computational requirements and the VM's capabilities.

Algorithm II shows the pseudo code for the first improvement of the MRFO algorithm by employing Lévy-flight technique to enhance the exploration of the algorithm.

Algorithm II: Lévy-Manta ray Algorithm (L-MRFOA).

MRFO Original initialization

.

.

//*Cyclone foraging*

.

.

ELSE // *Chain foraging*

// **First Improvement**: introduce Lévy flight mechanism for enhancing exploration by generating step sizes based on a *Lévy* distribution, introducing variability that allows the algorithm to take occasional large steps, improving its ability to explore the search space and avoid local minima.

Leves = Lévy ()

If $i = 1$ THEN

$$x_{i(t+1)} = x_{i(t)} + levies\,[i] * \left(x_{best(t)} - x_{i(t)}\right) + \alpha * \left(x_{best(t)} - x_{i(t)}\right)$$

ELSE

$$x_{i(t+1)} = x_{i(t)} + levies\,[i] * \left(x_{i-1(t)} - x_{i(t)}\right) + \alpha * \left(x_{best(t)} - x_{i(t)}\right)$$

END IF

END IF.

// Ensure the new positions are within search space boundaries

$$x_i(t+1) = Bounds\; x_{i(t+1)}, x_l, x_u)$$

Compute the fitness of each individual $f(x_{i(t+1)})$

$IF\ \ f(x_{i(t+1)}) < f(x_{best})$

THEN $x_{best} = x_{i(t+1)}$

END IF

END FOR

// *Somersault foraging*

FOR i=1 TO *N* DO

$$x_{i(t+1)} = x_{i(t)} + S * \left(r_2 * x_{best} - r_3 * x_{i(t)}\right)$$

// Ensure the new positions are within search space boundaries

$$x_{i(t+1)} = Bound\; x_{i(t+1)}, x_l, x_u)$$

Compute the fitness of each individual $f(x_{i(t+1)})$

$IF\ \ f(x_{i(t+1)}) < f(x_{best})$

THEN $x_{best} = x_{i(t+1)}$

END IF

END FOR.

END WHILE.

Return the best solution found so far x_{best}

After calculating these values, the algorithm identifies the VM that offers the shortest execution time for the task. The task is then assigned to the VM with the optimal performance. After assigning the tasks, the overall effectiveness of the current solution is evaluated, considering both the execution time and cost. If the new allocation improves the solution, it is retained; otherwise, the algorithm reverts to the previous allocation. Finally, once all tasks have been processed, whether independent or dependent, the fitness of the

current solution is recalculated to ensure that only the best task-to-VM allocation is preserved. H-MRFOA, as illustrated in the results below in the Pseudo code

Algorithm III: Heuristic Manta ray Algorithm (H-MRFOA).

MRFO Original initialization
.
.
//*Cyclone foraging*
.
.
// *Chain foraging*
.
.
// *Somersault foraging*
.
.
Return the best solution found so far x_{best}
// **Second Improvement**: heuristic dependency management, tackling the dependency for tasks that allocated among available virtual machines (VMs)Initialize task list, available VMs, and other necessary parameters in work environment
Set best solution *x_best* to the best solution found so far
FOR each task*i* in *taskList* **DO**
// Check for dependencies (*parent tasks*)
// Store dependent tasks temporarily
FOR each *parent* of task*i* **DO**
Add parent to *parentList*
END FOR
// For each parent task, optimize resource allocation
FOR each *parent* in parentList **DO**
// Initialize best allocation for current task
Copy current best solution *x_best* to t_best
fitness = calculate Fitness(x_*best*)
// Calculate execution time for each VM
FOR each vm in available VMs **DO**
MET [vm]=Execution time of parent on vm
END FOR
// Find the VM with the shortest execution time
// Assign the parent task to the best VM
X_best[parent] = MIN(MET)
// Check if the new allocation improves the solution
IF calculate fitness(x_*best*) < fitness **THEN**
// Keep the new allocation
continue for next parent task
ELSE
// Revert to the previous allocation
retrieve previous best solution x_*best* to *t_best*
END IF
END FOR
END FOR

// Return the best solution found so far Return the best solution *x_best*

The final stage of optimization leverages a hybrid approach, combining two powerful techniques: Lévy flight and Heuristic Dependency Management (HDM). This strategy aims to balance global exploration and local exploitation in the search for an optimal solution. Lévy flight enables the algorithm to perform large jumps in the solution space, helping avoid local optima by exploring new regions that incremental steps might miss. Meanwhile, HDM enhances the algorithm's ability to refine these potential solutions, particularly in optimizing task dependencies. By incorporating Lévy flight, the algorithm gains the ability to cover diverse areas of the solution space using a probability distribution (Lévy distribution) that favors small steps but allows for large leaps. This structured randomness ensures that unexplored areas are efficiently reached, improving the chances of finding a globally optimal solution. On the other hand, HDM plays a crucial role in the local exploitation phase. Once promising solutions are identified, HDM fine-tunes them by analyzing task dependencies and VM loads, ensuring that tasks are assigned to minimize overall completion time. This practical optimization is essential for improving real-world performance in distributed systems.

The integration of HDM as a second improvement ensures that solutions are not only theoretically optimal but also practical in execution. It enhances the algorithm's ability to handle complex optimization problems where efficient resource management and task scheduling are key. In essence, this hybrid approach combines the exploratory power of Lévy flights with the precise, task-optimized benefits of HDM, making the algorithm more robust, adaptive, and capable of delivering high-performance results in real-world applications that resulting in a more comprehensive and effective optimization algorithm as illustrated in below Pseudo code for Lévy-Heuristic Manat ray Algorithm (LH-MRFOA).

Algorithm IV: Lévy-Heuristic Manat ray Algorithm (LH-MRFOA).
MRFO Original initialization . . . //*Cyclone foraging* // **First improvement**: introduce Lévy flight mechanism for enhancing exploration . . . // *Chain foraging* . . . // *Somersault foraging* Return the best solution found so far x_{best} // **Second improvement**: heuristic dependency management // Return the best solution found so far Return the best solution *x_best* END FUNCTION

5. Experimental Setup

5.1 Workflow Simulation Platform.

A workflow simulation platform (WFS) is essential for testing and enhancing job scheduling algorithms in research and academic settings, especially in cloud computing. This platform ensures that the findings are consistent, reproducible, and applicable to real-world settings by offering a controlled environment for the testing and comparing alternative scheduling techniques under diverse conditions [29]. WFS is designed to model and simulate the execution of workflows in distributed computing environments. These platforms allow researchers to emulate the behavior of complex workflows, which are often represented as Directed Acyclic Graphs (DAGs) [30]. Each node in the DAG represents a specific task, while the edges signify dependencies between these tasks. The primary goal of using a WFS is to analyze the performance of different scheduling algorithms, particularly in terms of key metrics such as makespan, processing cost, resource utilization, and energy consumption. In cloud computing, task scheduling is a complex problem due to the dynamic nature of cloud resources and the diverse requirements of scientific workflows. Workflow simulation platforms are particularly valuable in this domain because they allow for testing scheduling algorithms under different configurations of virtual machines, resource availability, and workload sizes. This is critical for understanding how algorithms perform under varying conditions, which is essential for optimizing cloud resource management. The WFS allows researchers to assess the algorithm's effectiveness in minimizing makespan and cost. By simulating different workload scenarios, including those with varying task complexities and interdependencies, the WFS can provide insights into the scalability and robustness of the proposed algorithms.

5.2 Virtual Machines (VMs) Setup

The task scheduling experiment was conducted using a workflow simulator configured to run on 15 virtual machines distributed among three different groups: slow, moderate, and fast. This grouping ensures a fair distribution of tasks across various types of VMs, reflecting a range of computational capabilities as illustrated in the table below (2). Available VMs are grouped into three categories based on their computational power. This allows the task scheduler to allocate tasks to VMs with appropriate resources, depending on the task's computational requirements. within each group, the MIPS (Million Instructions Per Second) value is randomly assigned within a specified range. This randomization simulates a more realistic cloud environment where VM capabilities can vary even within the same category. RAM and bandwidth are also set according to the VM group. The 'slow' group has lower RAM and MIPS values, while the 'fast' group has higher values, making it suitable for more computationally intensive tasks. The setup mimics a typical cloud environment where resources are heterogeneous by ensuring that tasks are distributed among VMs with different capabilities. This helps to test the robustness of the scheduling algorithm across various conditions. Using WFS provides a controlled environment for simulating the task scheduling process, making it possible to evaluate the effectiveness of the proposed algorithm under different VM configurations. This setup is essential for evaluating the performance of the task scheduling algorithm in a realistic cloud computing environment, ensuring that it can effectively manage and distribute workloads across VMs with varying capabilities.

Table 2: Specifications of Virtual Machines Used in Workflow Simulations

Group#	# of VMs	RAM	Range	MIPS
G1: Slow	5	512	(0-4)	1000-3000
G2: Moderate	5	512	(5-9)	3000-6000
G3: Fast	5	1024	(10-14)	6000-10000

5.3 Experimental Process

We developed a run processing system within our workflow simulator that executes each algorithm ten times to evaluate makespan and processing cost using a weighted summation formula. The comparison conducted among (**MRFOA**) and its updated versions Lévy-Manta ray Foraging Algorithm (**L-MRFOA**), Heuristic-Manta ray Foraging Algorithm (**H-MRFOA**), Lévy Heuristic-Manta ray Foraging Algorithm (**LH-MRFOA**) and the top of bio-inspired algorithms (**GA**) and (**PSO**), by using five standard data sets fig (1) with different scales and considering the average of ten times of the whole run-in process of workflow simulator table (2), table (3) and table (4).

6. Results and Discussion

6.1 Empirical Results

The empirical results offer a thorough evaluation of the performance characteristics of the algorithms under study, focusing on key metrics such as makespan, cost, and fitness. These metrics provide essential insights into how each algorithm manages task scheduling within complex cloud computing environments. Through rigorous testing across diverse workloads. This section delves into the performance test's quantitative outcomes, highlighting each algorithm's strengths and weaknesses.

The results reveal significant distinctions in how these algorithms handle various scheduling challenges, with LH-MRFOA frequently outperforming other algorithms, GA and PSO. The findings underscore the robust capability of LH-MRFOA to optimize resource allocation, minimize execution time, and reduce operational costs, making it an optimal solution for dynamic and large-scale cloud environments. In terms of makespan, the LH-MRFOA consistently outperformed its counterparts, demonstrating a remarkable ability to minimize the total time required for task completion. Across various datasets, LH-MRFOA achieved significantly lower makespan values, particularly in scenarios involving large and complex workloads during testing makespan as shown in Table (3). This reduction in makespan is crucial in cloud computing environments, where time efficiency directly impacts overall system performance and user satisfaction. The algorithm's advanced heuristic strategies and robust search capabilities enable it to navigate and optimize scheduling tasks more effectively than traditional algorithms such as GA and PSO.

Table 3: Best Makespan

#	Data set	GA	PSO	MRFOA	L-MRFOA	H-MRFOA	LH-MRFOA
1.	Inspiral_30	3389.13	3107.70	2860.94	2976.31	2875.47	**2799.32**
2.	Inspiral_1000	255438.98	210553.27	236622.81	198138.36	115326.90	**101372.02**
3.	CyberShake_30	287.83	270.75	244.31	236.80	232.47	**218.99**
4.	CyberShake_1000	14872.72	14781.01	13621.18	13439.58	11222.14	**10860.59**
5.	Montage_25	245.08	216.51	231.44	223.30	217.71	**187.45**
6.	Montage_100	6301.10	5364.55	4830.51	4913.52	3761.60	**3752.01**
7.	Sipht_30	2767.21	2620.10	2684.59	2616.49	2612.90	**2612.01**
8.	Sipht_1000	34034.23	32856.94	27482.56	32199.65	20732.19	**19652.69**
9.	Epigenomics_24	2375.01	2580.19	2363.02	2360.27	2238.06	**2146.30**
10	Epigenomics_997	973484.72	882633.77	856202.79	824748.70	410740.87	**398818.30**

Turning to the cost, the LH-MRFOA also proved highly effective in minimizing the operational expenses associated with task scheduling. Cost efficiency is a critical factor in cloud computing, where resource utilization directly translates to financial expenditure.
LH-MRFOA demonstrated a consistent ability to reduce costs across diverse workload scenarios, outperforming other algorithms by optimizing resource allocation and reducing the computational overhead, as illustrated in Table (4). This cost-effectiveness makes LH-MRFOA an attractive option for cloud service providers aiming to maximize profitability while maintaining high levels of performance. By achieving lower costs without compromising on efficiency or scalability, LH-MRFOA positions itself as a superior solution for managing the economic demands of cloud-based operations.

Table 4: Best Cost

#	Data set	GA	PSO	MRFOA	L-MRFOA	H-MRFOA	LH-MRFOA
11	Inspiral_30	647.65	632.94	583.29	575.21	572.24	**556.10**
12	Inspiral_1000	24430.67	24080.96	24036.85	22860.16	23443.54	**22301.88**
13	CyberShake_30	19839.65	19695.84	19687.56	19690.19	19646.01	**19645.61**
14	CyberShake_1000	106270.17	104056.90	102352.58	102785.97	94176.64	**93953.63**
15	Montage_25	135.08	136.16	125.93	129.67	124.77	**121.51**
16	Montage_100	576.67	588.74	535.21	540.06	542.31	**516.74**
17	Sipht_30	529.02	538.16	512.41	513.08	511.22	**506.60**
18	Sipht_1000	20810.20	20519.92	20954.26	21020.73	19976.43	**19729.46**
19	Epigenomics_24	3393.01	3229.67	3164.84	3228.88	3166.17	**3165.74**
20	Epigenomics_997	680448.43	635623.36	631217.41	584845.97	579503.43	**558601.39**

Finally, when considering fitness, which provides a comprehensive measure of an algorithm's overall performance, LH-MRFOA once again demonstrated its superiority. Fitness encapsulates various aspects of task scheduling, including the balance between exploration and exploitation, efficiency in resource use, and the ability to adapt to varying workloads. LH-MRFOA consistently achieved higher fitness scores across different datasets, reflecting its robust capability to optimize multiple performance criteria simultaneously, as illustrated below in Table (5). This high fitness indicates that LH-MRFOA is not only effective in specific metrics like makespan and cost but is also versatile enough to maintain strong performance across a range of conditions. This makes LH-MRFOA a highly reliable choice for real-world applications, where diverse and dynamic cloud environments demand an algorithm that can consistently deliver optimal results.

Table 5: Best Fitness

#	Data set	GA	PSO	MRFOA	L-MRFOA	H-MRFOA	LH-MRFOA
21	Inspiral_30	2176.51	1865.10	1883.92	1933.19	1832.15	**1800.98**
22	Inspiral_1000	143684.31	123370.91	119832.70	115946.63	68411.38	**66834.45**
23	CyberShake_30	10040.82	9994.98	9930.06	9932.40	9932.80	**9928.21**
24	CyberShake_1000	60581.67	57353.46	56657.83	56988.78	52021.07	**51923.83**
25	Montage_25	213.82	197.64	190.03	200.59	180.44	**176.41**
26	Montage_100	3575.65	3168.25	3052.49	2893.94	2264.96	**2214.69**
27	Sipht_30	1750.57	1667.66	1664.87	1665.31	1550.08	**1541.81**
28	Sipht_1000	27606.74	25750.23	26657.38	25113.37	20247.89	**19201.83**
29	Epigenomics_24	2980.56	2856.72	2791.28	2755.68	2761.34	**2661.55**
30	Epigenomics_997	908657.14	801416.80	781958.68	738252.03	523809.04	**492700.21**

6.2 Performance Variability Analysis

Task scheduling in the cloud should be very unpredictable in nature. While observing the facts, the prior result proved that GA was found to be performing variably over the datasets. For instance, with the best cost scenario in GA, it was found to be constant in performing well at low iterations but losing hold in the latter iterations against LH-MRFOA. This trend can be seen in the best cost in Fig (7), where LH-MRFOA clearly had better performance as the algorithm continued, with better lower costs from further iterations.

On the other hand, as shown in Fig (8) best makespan, GA had consistent performance across the iteration but was greatly surpassed by the LH-MRFOA, mainly in the minimizing makespan measure. The LH-MRFOA not only showed superiority in the reduction of makespan but also in the improved consistency across further iterations. Similarly, from the best fitness analysis Fig (8), GA remained strong in performance at the start, and it was only later surpassed by both the L-HMRFOA and LH-MRFOA algorithms, especially as alternating count increased. There are apparent fluctuations in the outcome due to the variability necessary to select the algorithm, which then counts with the explicit nature of the workload. Different scenarios expose the strengths and weaknesses of each algorithm. The LH-MRFOA showed the best performance regarding all the applied criteria; this was carried out due to its hybrid nature. Therefore, it is a generalized approach that can underpin any cloud computing activity.

6.3 Scalability Analysis

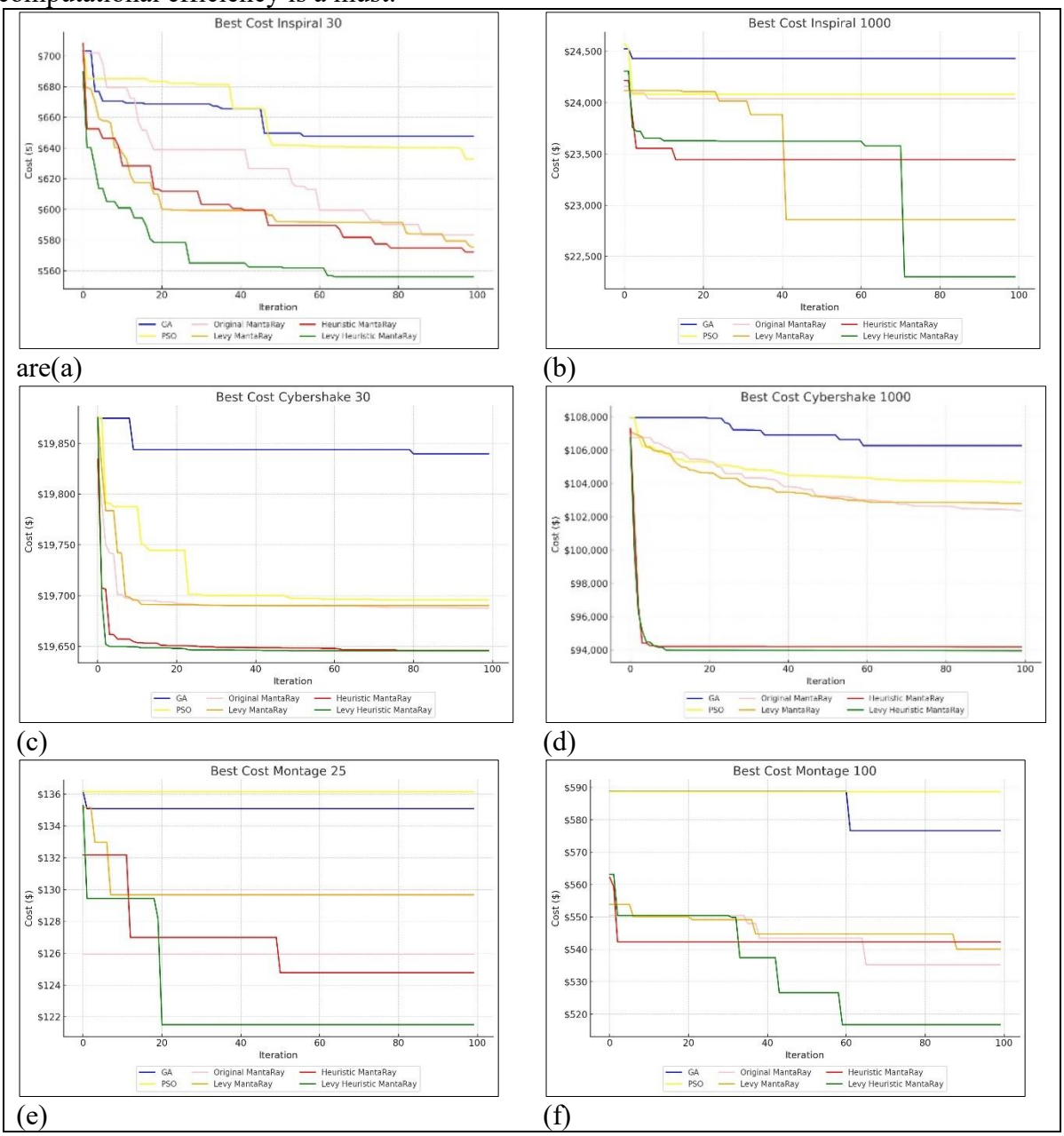
Adaptive scalability regarding the available resources to be provisioned forms the most critical issue in the task scheduling algorithm. This becomes acute in nature for the cloud computing system in which resource adaptability and dynamic provisioning are applied in relation to the change of workload. The results obtained from the three different applied criteria, designated as best cost, best fitness, and best makespan were used in scaling the algorithm's behavior regarding L-MRFOA and its heuristic extension. These algorithms were not only holding their performance but further improving as the iteration counts went up, thus showing an adaptability to increasing workload sizes. GA was competitive in the lower initial stages or with lower iterations but generally struggled in the best makespan and best fitness figures, where GA predictably leveled off in performance or even declined in the face of increasing workload size, while the LH-MRFOA continued to optimize effectively. These results show that the scalability of LH-MRFOA is a better fit for a larger and more dynamic cloud environment.

6.4 Consistency Evaluation

An algorithm representative of its reliability and stability is consistent throughout varied workloads. LH-MRFOA, according to the results obtained on the best cost, best fitness, and best makespan figures, is proven to be reasonably consistent throughout varied situations. It outperformed other algorithms, like GA and PSO, for different iteration numbers, as seen below. The algorithm's consistency indicates that it is powerful and effective enough to handle diversified task scheduling scenarios and therefore clearly establishes its candidature for real-world deployment in a cloud environment. However, it is also evident from these figures that in the case of extreme workloads, represented by the best makespan dataset, the performance gaps between algorithms became more pronounced. The effectiveness of the LH-MRFOA is seen in its ability to hold performance by retaining robustness in resource allocation frameworks for a high-demand cloud environment.

6.5 Complexity vs. Performance Trade-off

The performance of complex algorithms, e.g., L-MRFOA and its heuristic variant, versus simpler algorithms, e.g., GA and PSO, tends to be proportional to workload or iteration count; see the figures. It is not universal; at a good cost Fig.8, GA and PSO remain competitive in the earlier iterations, meaning that an increase in algorithmic complexity does not always bring better performance. This indicates the necessity of a delicate evaluation that will consider both complexity and performance outcomes while selecting the right approach to be used specifically in scheduling tasks for the cloud environment. Whereas this LH-MRFOA offers huge competitive advantages in most scenarios, at the cost of performance, some simpler algorithms will remain good enough in the simpler contexts or when computational efficiency is a must.



are(a) (b)

(c) (d)

(e) (f)

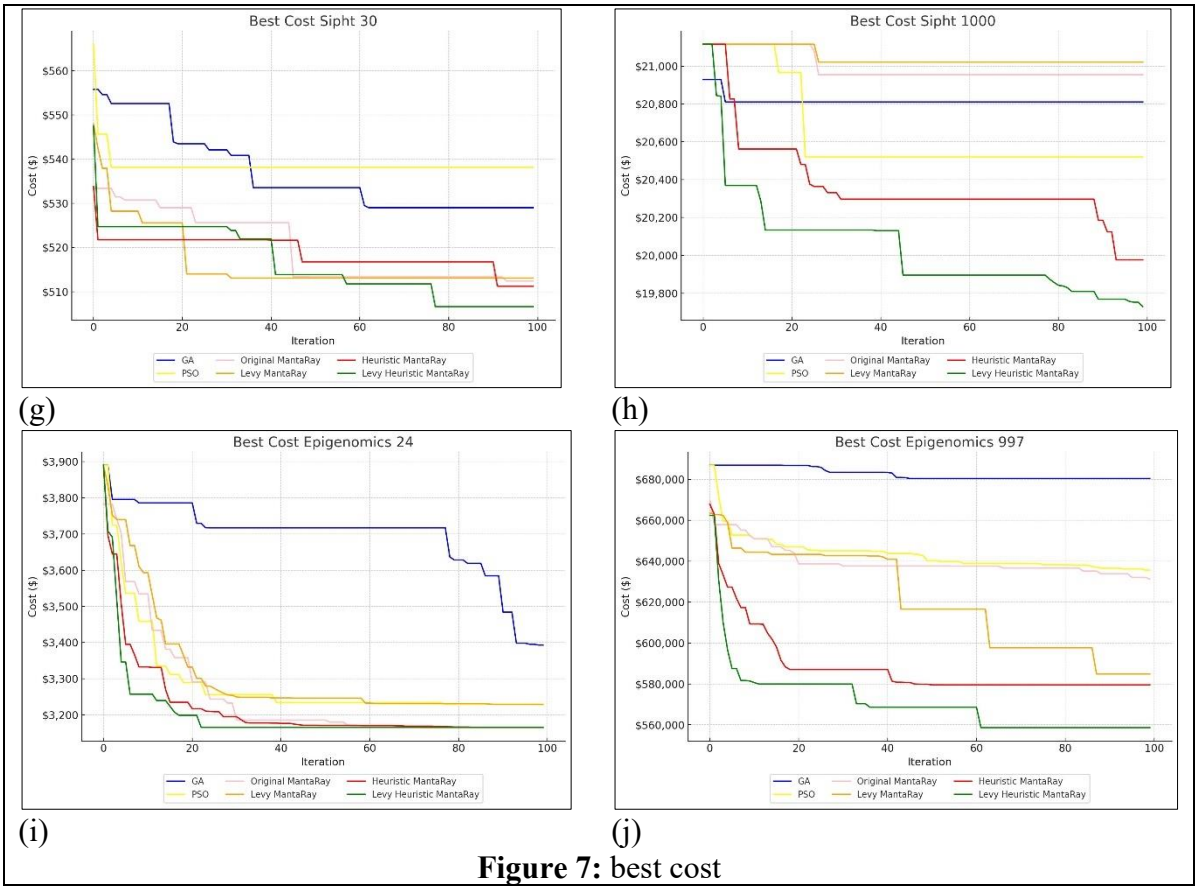
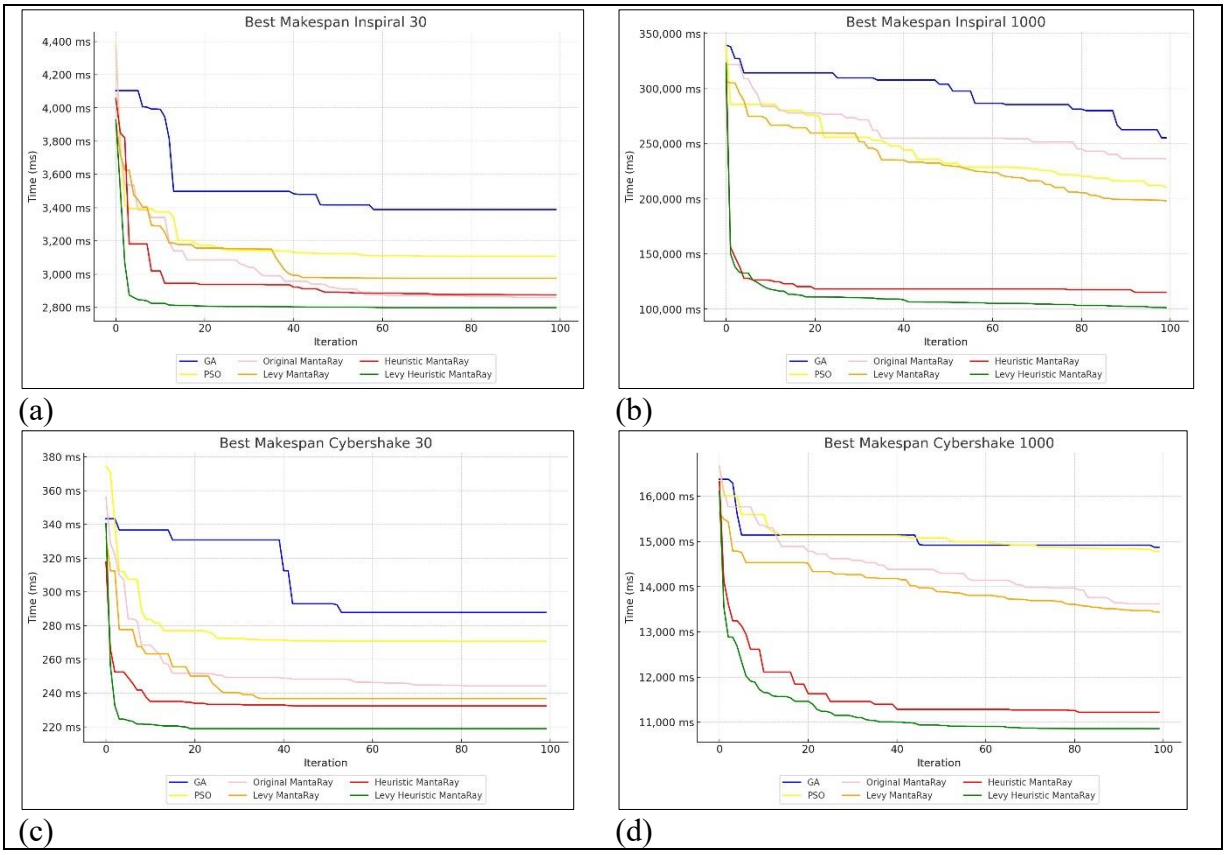
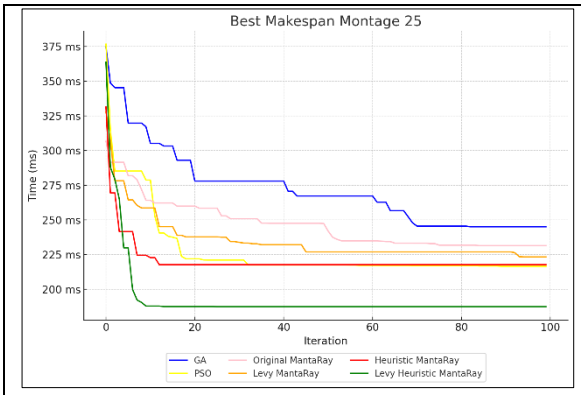
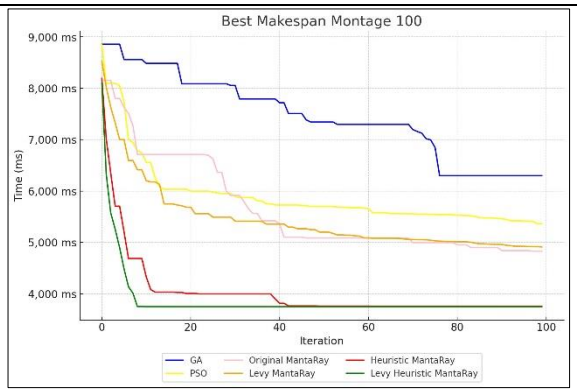


Figure 7: best cost

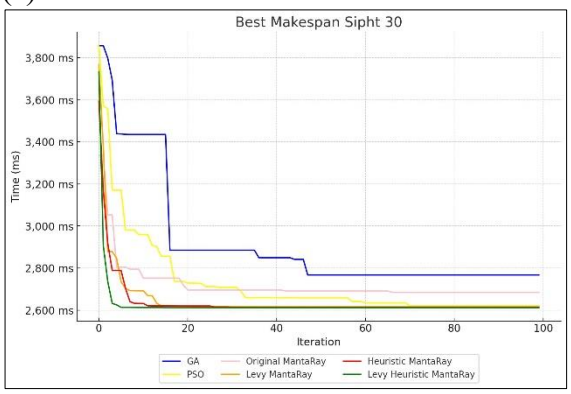




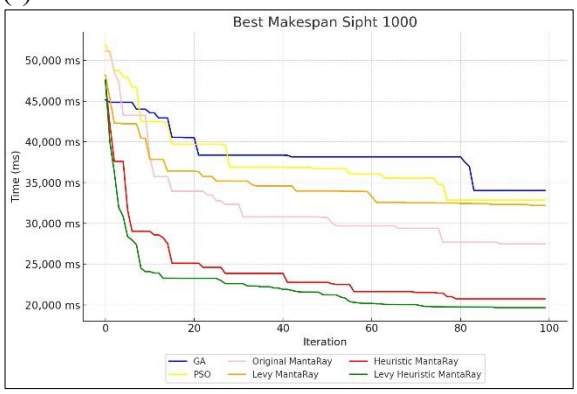
(e)



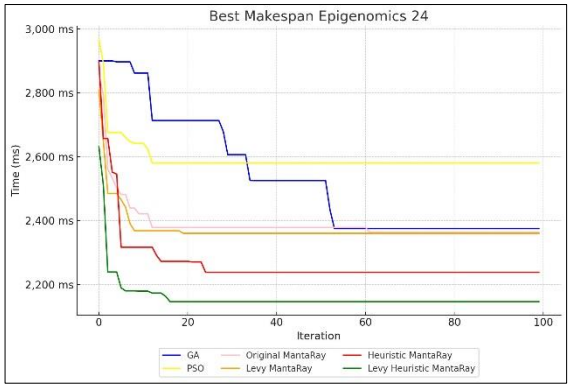
(f)



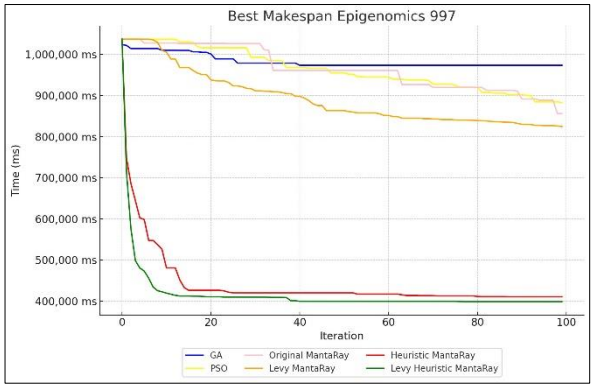
(g)



(h)

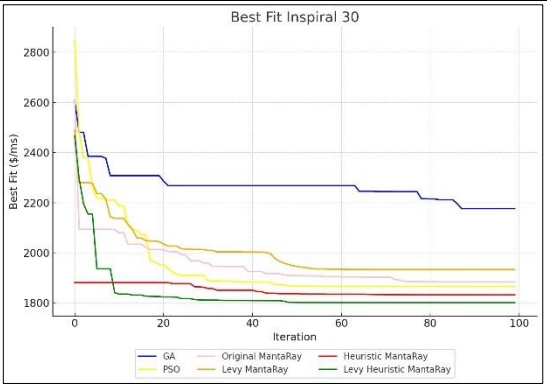


(i)

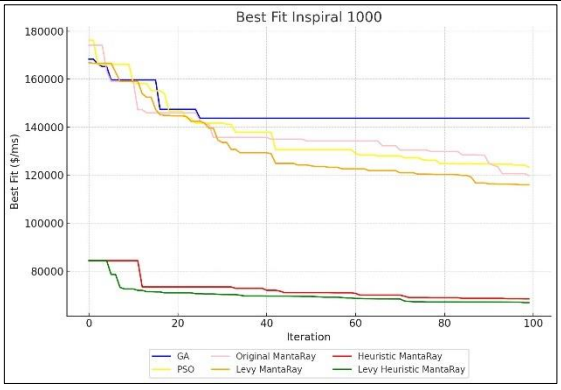


(j)

Fig (8) best makespan



(a)



(b)

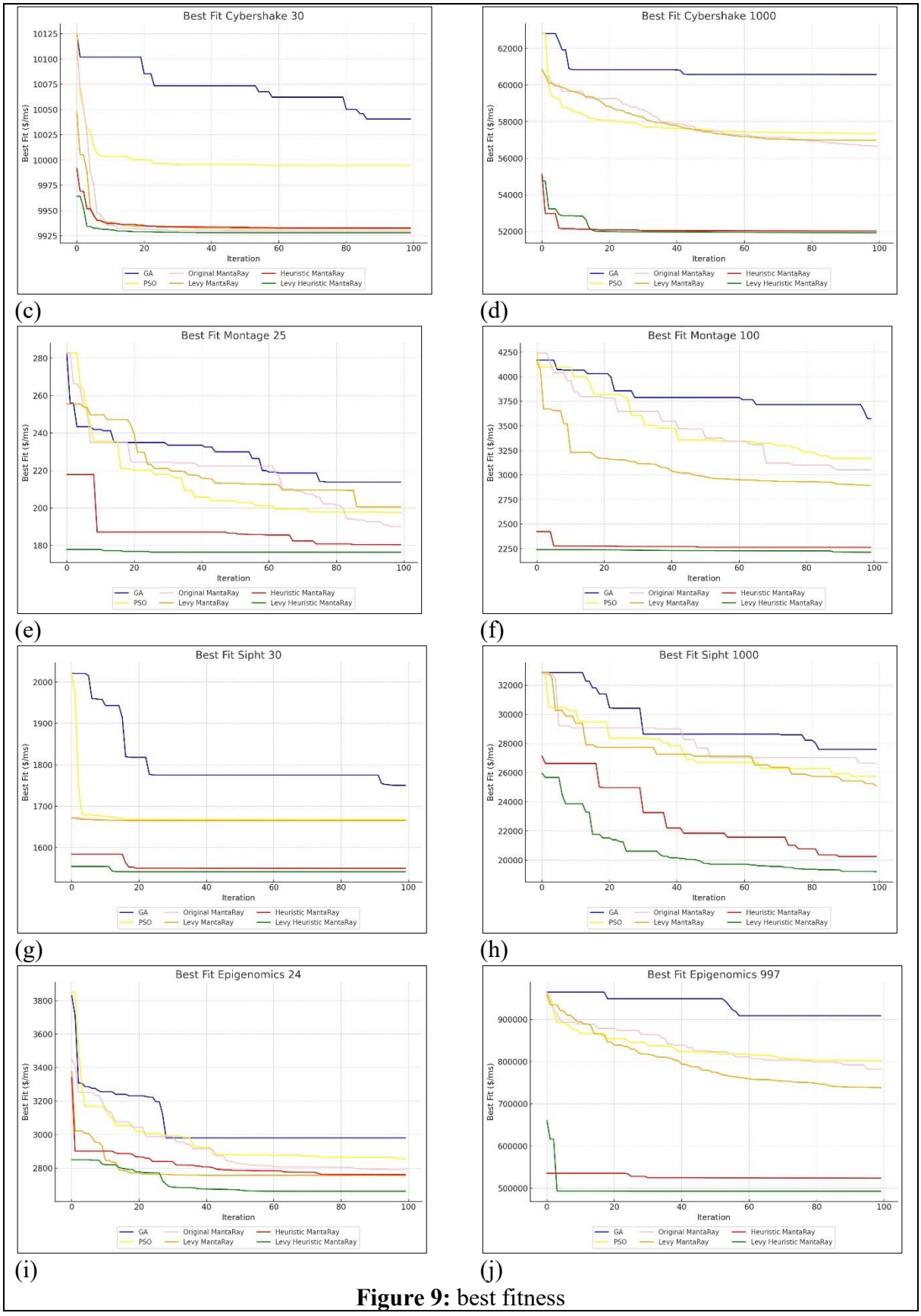


Figure 9: best fitness

6.6 Discussion

The LH-MRFOA is a potent step forward in optimization task scheduling for cloud computing. With the utilization of the Lévy flight search strategies in optimizing landscapes,

LH-MRFOA has shown a better balance between exploitation and exploration. The results shown in the best cost, Best makespan, and best fitness figures show that the LH-MRFOA outperform standard bio-inspired algorithms like GA and PSO in most cases, especially when higher scalability and consistency are required. Among all, the LH-MRFOA especially can scale up to an increased iteration count or workload size where it either maintained or improved performance compared to the other algorithms that failed. The experimental platform, constructed with rich configurations in virtual machines, represents more precisely the actual cloud environment's complexity and makes the research outcome more applicable. The scalability spoken about by LH-MRFOA speaks to its overall suitability for workload conditions with increased magnitude without degradation of performance and, as such, is an ideal candidate for dynamic, changing cloud computing situations.

7. Conclusion

This work has fully investigated the efficiency and effectiveness of different task scheduling algorithms within the cloud computing environment, emphasizing the LH-MRFOA. The analysis has dealt with some critical issues such as performance variability, scalability, consistency, and the complexity-performance trade-off, and arrived at some significant results. The final results underline the subtleties of task scheduling optimization since the performance of an algorithm varies quite significantly according to the workload. Another nature-based algorithm, like GA and PSO, has shown initial efficiencies, especially under scenarios with less workload. Advanced methods like LH-MRFOA were consistently outperforming as the workload increased with the increase in both the aspects of size and complexity. Furthermore, the superiority of these techniques is highly visible for later iterations as well, where the LH-MRFOA outperforms, especially its Lévy Heuristic variant, in minimizing the cost, optimizing the fitness, and reducing makespan. Results of the scalability analysis position LH-MRFOA at good adaptability to various workload demands since it can utilize properly available resources and optimize an assignment of tasks in a dynamic cloud environment. In contrast, GA and PSO failed in scalability for more complex scenarios, where LH-MRFOA kept its performance constant or even improved when the number of iterations and the size of workloads increased. Real-world cloud computing applications greatly require this adaptability since the demand for resources is constantly changing. Consistency evaluation further proved the reliability and robustness of LH-MRFOA. Regardless of scheduling scenarios of diverse tasks, LH-MRFOA performs well when it uses Lévy flight strategies compared with other algorithms. This consistency across different metrics/iterations further confirms that LH-MRFOA has the potential to be applied effectively to real-world problems by capturing the deployment challenges of cloud resource management.

Moreover, this work has also outlined one significant trade-off between the complexity of an algorithm and its performance. While more complex algorithms, such as LH-MRFOA, often outperformed simpler variants, especially for large and iteratively solved scenarios, there were cases when GA and PSO remained competitive. This henceforth explains why a delicate approach in the evaluation criteria, such as adaptability and optimization of efficiency in resource utilization through scalability, has to be applied to task scheduling algorithms for the cloud environment. In summary, the LH-MRFOA is one approach essential for task scheduling optimization, which outperforms existing ones in improved performance, scalability, and reliability. These features become necessary in dealing with complex and dynamic workloads consistently for different metrics and render it a viable solution for managing cloud resources. Further research can be conducted to enhance LH-MRFOA further and apply it in other optimization areas related to cloud computing to continuously improve cloud resource management and optimization techniques. Therefore, this research contributes to the ever-growing scientific knowledge base on improving efficiency and

effectiveness in task scheduling within cloud computing to enhance overall performance and reliability in cloud-based systems.

References

[1] M. F. Younis, "Enhancing cloud resource management based on intelligent system," *Baghdad Science Journal*, vol. 21, no. 6, pp. 2156–2156, 2024.

[2] D. R. Abdulrazzaq, N. M. Shati, and H. K. Hoomod, "Task scheduling in a cloud environment based on meta-heuristic approaches: A survey," *Iraqi Journal of Science*, vol. 65, no. 2, pp. 1001–1023, 2024.

[3] J. Yu, R. Buyya, and C. K. Tham, "Cost-based scheduling of scientific workflow applications on utility grids," in *Proc. First Int. Conf. e-Science Grid Comput. (e-Science'05)*, Melbourne, VIC, Australia, Dec. 2005, pp. 8–147, doi: 10.1109/E-SCIENCE.2005.26.

[4] W. Viriyasitavat, L. D. Xu, G. Dhiman, A. Sapsomboon, V. Pungpapong, and Z. Bi, "Service workflow: State-of-the-art and future trends," *IEEE Transactions on Services Computing*, vol. 16, no. 1, pp. 757–772, 2021.

[5] H. Kchaou, Z. Kechaou, and A. M. Alimi, "A PSO task scheduling and IT2FCM fuzzy data placement strategy for scientific cloud workflows," *Journal of Computational Science*, vol. 64, p. 101840, 2022.

[6] I. A. Abduljabbar and S. M. Abdullah, "An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem," *Baghdad Science Journal*, vol. 19, no. 2, pp. 399-408, Apr. 2022. DOI: 10.21123/bsj.2022.19.2.0399.

[7] Y. Zhang, L. Wu, M. Li, T. Zhao, and X. Cai, "Dynamic multi-objective workflow scheduling for combined resources in cloud," *Simulation Modelling Practice and Theory*, vol. 129, p. 102835, 2023.

[8] A. Talha, A. Bouayad, and M. O. C. Malki, "An improved pathfinder algorithm using opposition-based learning for tasks scheduling in cloud environment," *Journal of Computational Science*, vol. 64, p. 101873, 2022.

[9] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization-based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Computer Communications*, vol. 187, pp. 35–44, 2022.

[10] H. Hafsi, H. Gharsellaoui, and S. Bouamama, "Genetically-modified multi-objective particle swarm optimization approach for high-performance computing workflow scheduling," *Applied Soft Computing*, vol. 122, p. 108791, 2022.

[11] J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, "Cost and makespan-aware workflow scheduling in hybrid clouds," *Journal of Systems Architecture*, vol. 100, p. 101631, 2019.

[12] B. H. Abed-Alguni and N. A. Alawad, "Distributed Grey Heuristic for scheduling of workflow applications in cloud environments," *Applied Soft Computing*, vol. 102, p. 107113, 2021.

[13] J. K. Konjaang and L. Xu, "Multi-objective workflow optimization strategy (MOWOS) for cloud computing," *Journal of Cloud Computing*, vol. 10, no. 1, p. 11, 2021.

[14] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2020.

[15] H. Aziza and S. Krichen, "A hybrid genetic algorithm for scientific workflow scheduling in cloud environment," *Neural Computing and Applications*, vol. 32, no. 18, pp. 15263–15278, 2020.

[16] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling," *Cluster Computing*, vol. 24, pp. 1479–1503, 2021.

[17] N. Anwar and H. Deng, "A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment," *Applied Sciences*, vol. 8, no. 4, p. 538, 2018.

[18] S. Kumar and R. Buyya, "Green cloud computing and environmental sustainability," in *Harnessing Green IT: Principles and Practices*, pp. 315–339, 2012.

[19] M. A. Rodriguez and R. Buyya, "Deadline-based resource provisioning and scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2014.

[20] W. Zhao, Z. Zhang, and L. Wang, "Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications," *Engineering Applications of Artificial Intelligence*, vol. 87, p. 103300, 2020.

[21] H. Dewar, P. Mous, M. Domeier, A. Muljadi, J. Pet, and J. Whitty, "Movements and site fidelity of the giant manta ray, Manta birostris, in the Komodo Marine Park, Indonesia," *Marine Biology*, vol. 155, pp. 121–133, 2008.

[22] U. C. Ben, A. E. Akpan, C. C. Mbonu, and E. D. Ebong, "Novel methodology for interpretation of magnetic anomalies due to two-dimensional dipping dikes using the manta ray foraging optimization," *Journal of Applied Geophysics*, vol. 192, p. 104405, 2021.

[23] A. M. Shaheen, A. R. Ginidi, R. A. El-Sehiemy, and S. S. Ghoneim, "Economic power and heat dispatch in cogeneration energy systems using manta ray foraging optimizer," *IEEE Access*, vol. 8, pp. 208281–208295, 2020.

[24] E. H. Houssein, I. E. Ibrahim, N. Neggaz, M. Hassaballah, and Y. M. Wazery, "An efficient ECG arrhythmia classification method based on manta ray foraging optimization," *Expert Systems with Applications*, vol. 181, p. 115131, 2021.

[25] S. Duman, A. Dalcalı, and H. Özbay, "Manta ray foraging optimization algorithm–based feedforward neural network for electric energy consumption forecasting," *International Transactions on Electrical Energy Systems*, vol. 31, no. 9, p. e12999, 2021.

[26] S. A. Alsaidy and N. A. Abdullah, "Power-efficient virtual machine placement in cloud datacenters using heuristic-assisted enhanced discrete particle swarm optimization," *Iraqi Journal of Science*, vol. 63, no. 10, pp. 4499–4517, 2022.

[27] Y. Liao, W. Zhao, and L. Wang, "Improved manta ray foraging optimization for parameters identification of magnetorheological dampers," *Mathematics*, vol. 9, no. 18, p. 2230, 2021.

[28] M. P. Ćalasan, A. Jovanović, V. Rubežić, D. Mujičić, and A. Deriszadeh, "Notes on parameter estimation for single-phase transformer," *IEEE Transactions on Industry Applications*, vol. 56, no. 4, pp. 3710–3718, 2020.

[29] H. S. Ramadan and A. M. Helmi, "Optimal reconfiguration for vulnerable radial smart grids under uncertain operating conditions," *Computers & Electrical Engineering*, vol. 93, p. 107310, 2021.

[30] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," in *Proc. 2012 IEEE 8th Int. Conf. e-Science*, Chicago, IL, USA, Oct. 2012, pp. 1–8, doi: 10.1109/eScience.2012.6404430.