Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44) 2** : 1–16.
ISSN-0037-8712
doi:10.5269/bspm.79519

Multi-Objective Dependent Tasks Scheduling Using AWGWO Algorithm in Cloud Computing

Chotu Lal * and Harish Sharma

ABSTRACT: This paper introduces a method to solve workflow scheduling problems in clouds. The workflow scheduling is an appropriate scheduling to reduce the total execution time (TET) and total execution cost (TEC) of tasks when tasks are interdependent. The nature of workflow scheduling is NP- hard, so traditional algorithms have failed to solve it. This article proposes a meta-heuristic algorithm named adaptive weights grey wolf optimization (AWGWO) algorithm. The AWGWO is a new variant of GWO. This new variant is modified in two ways: first, considering the new wolf leader's name, gamma, and second, the weights assigned to wolf leaders are adaptive instead of the average weight of wolf leaders, similar to the original GWO algorithm. This modification of GWO maintains a balance between exploration and exploitation because the early phase describes exploration, while the latter phase describes exploitation. This balance enables efficient assignment of the tasks to virtual machines and reduces the TET and TEC of workflows. The proposed algorithm outperforms the ACO and GWO algorithms.

Keywords: Optimization algorithms, metaheuristic algorithms, adaptive weights grey wolf optimization, cloud computing, workflow scheduling.

Contents

1. Introduction

A workflow application represents an application that is based on the interdependent task [1,2]. Cloud computing system refers to a network of remote servers accessible via the Internet that provides services such as storage, management, and data processing [3,4,5]. The workflow has capabilities to execute the application, which are based on workflows, such as bioinformatics, astronomy, and physics [6]. Workflow scheduling is a process that assigns tasks of a workflow to virtual machines (VM), particularly when tasks are interdependent, to minimize resource usage costs. The workflow scheduling constitutes an NP-hard problem within cloud environments [7]. The workflow consists of interdependent tasks and is denoted by a directed acyclic graph (DAG). A DAG is represented as G(T, E), where each node T corresponds to a task and each edge E denotes dependencies between tasks.

* Corresponding author.
2020 *Mathematics Subject Classification:* 90B35.
Submitted October 13, 2025. Published April 17, 2026

Figure 1 illustrates a sample of a DAG, here L represents the level. At level 0, T1 is an entry, and at level 5, T9 and T10 are the exit tasks. The entry task denotes that which does not have a parent task. The exit task refers to a task that has no child tasks. The task T2 is present at level 1. The tasks T3 and T4 are represented at level 2. The tasks T5 and T6 are located at Level 3, while the tasks T7 and T8 are at Level 4, respectively.

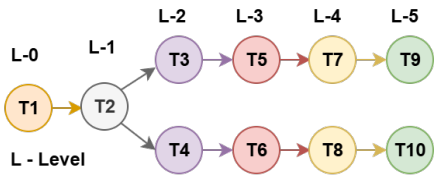

Figure 1: A sample of DAG

Figure 2 illustrates a DAG with mapping. The task T1 is allocated to VM3 at PM2, task T2 assigned to VM5 at PM1, task T3 given to VM2 at PM2, Task T4 assigned to VM4 at PM2, task T5 allocated to VM1 at PM1, task T6 assigned to VM5, at PM2, task T7 assigned to VM4 at PM2, task T8 and T9 are assigned to VM2 and VM3, respectively at PM1, and task T10 is assigned to VM1 at PM2.

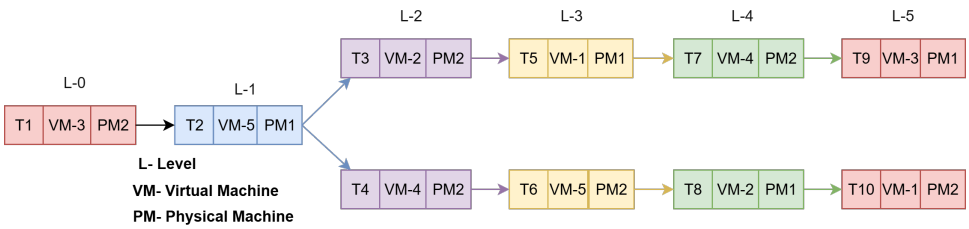

Figure 2: A DAG with mapping

In recent years, many researchers have proposed several scheduling algorithms, categorized into three categories: Traditional algorithms, heuristic algorithms, and metaheuristic algorithms. Traditional scheduling methods, such as first come first serve (FCFS) [10], round robin (RR) [9], and Min-Min 2014 [11] or Max-Min [12], are simple and easy to use; however, they are not suitable for cloud computing because they cannot handle heterogeneous resources and large workflow sizes. These methods often cause unbalanced load, longer execution times, and higher costs. Heuristic algorithms like heterogeneous earliest finish time (HEFT) [13] and critical path on a processor (CPOP) [13] give the best results by using task priorities. These algorithms are problem-specific and do not always guarantee the best solution in large-scale and dynamic cloud environments. Metaheuristic algorithms for examples particle swarm optimization (PSO) [14], PSO with load balancing mutation [15], pareto-based grey wolf optimizer (PGWO) [16], multi-objective ant colony system (MOACS) [17], hybrid gravitational search algorithm (GSA) [18], genetic algorithm (GA) with an adaptive penalty function [19]. These algorithms have balanced between exploration and exploitation. Exploration helps the search process move towards unexplored regions of the solution space, whereas exploitation works on improving the quality of the solutions that are already promising. These algorithms are widely applied for workflow scheduling in cloud systems. But even metaheuristics face limitations like premature convergence and getting stuck in local optima, which reduces their efficiency.

The grey wolf optimizer (GWO) [20] represents a balance between exploration and exploitation capabilities. Searching for prey represents the exploration capabilities, while hunting denotes the exploitation capabilities. This algorithm denotes balance amid exploration and exploitation through the guidance of α, β, and δ leaders, making it effective in achieving near-optimal task assignments.

The paper [21] presents GWO-based scheduling methods that improve cost and time. Sometimes it struggles with early convergence. The paper [22] represents the improved GWO approach. This

Table 1: Symbol table

Symbol	**Definition**	**Symbol**	**Definition**
DAG	Directed Acyclic Graph	RAM	Random Access Memory
T	Task	E	Edge
G	Graph	T^C	Completion Time
PM	Physical Machine	VM	Virtual machine
TET	Total Execution Time	TEC	Total Execution Cost
ω	Weight	WT	Waiting Time
ET	Execution Time	pred	Predecessor
LST	Least Start Time	$Size_{Task}$	Task Size
$num(PE)$	Number of cores of the virtual machines	PE_{Unit}	Processing capacity per core
LET	Least Execution Time	L	Level

paper discusses scheduling methods with crossover and mutation that improve adaptability but still face challenges of parameter sensitivity and increased computational cost in large-scale environments. The paper [8] illustrates a multi-objective task scheduling using the GWO approach. This method is used to minimize the makespan and cost in the cloud system. This paper [38] is used to reduce cost and time for the workflow scheduling problem. The paper [36] aims to reduce cost and execution time across four scientific workflows. The paper [39] focuses on reducing cost across three scientific workflows. The paper [40] aims to reduce total execution cost and execution time across four scientific workflows.

The study of literature review shows that the metaheuristic algorithm performs effectively in solving interdependent task scheduling problems. The metaheuristics approaches face challenges, such as premature convergence and getting trapped in local optima, which minimize their efficiency. To solve these limitations, researchers are improving the intelligence of standard metaheuristic algorithms by enhancing their exploration and exploitation abilities, so that they can escape local traps, search more effectively, and reach better solutions faster.

In this paper, we present the adaptive weights grey wolf optimization (AWGWO) algorithm for workflow scheduling, aiming to reduce total execution time (TET) and total execution cost (TEC). The proposed AWGWO improves the basic GWO by introducing a fourth leader, called Gamma, alongside Alpha, Beta, and Delta. This new leader also participates in guiding the process to search for prey in different directions, increasing the diversity of solutions and minimizing the probability of getting stuck in local optima. Additionally, the adaptive weighting mechanism is included to dynamically adjust the weight of each leader throughout the iterations. There is a balance between exploration and exploitation because exploration occurs in the early phase and exploitation in the later phase. As a result, AWGWO achieves better performance in finding optimal scheduling solutions compared to the basic GWO.

The performance assessment of the proposed AWGWO algorithm is used on the following five well-known scientific workflows: Montage [23], SIPHT [24], CyberShake [25], Inspiral [28], and Epigenomics [29]. These workflows were selected because they cover different scientific domains and vary significantly in task volume and computational requirements. The proposed algorithm is used to reduce TEC and TET for scientific workflows. The experiments were carried out using WorkflowSim [30], a simulation toolkit commonly used in workflow scheduling studies. Experimental evaluations demonstrate that the proposed AWGWO outperforms ant colony optimization (ACO) and standard GWO, achieving superior results in both TET and TEC. Table 1 presents the list of symbols used in this research paper along with their corresponding definitions.

2. Concept of Workflow Scheduling

This section explains how workflow scheduling is described using mathematical terms.

2.1. Mathematical Formulation of Workflow Scheduling

The objective function specifies the desired outcomes to be optimized using the proposed scheduling approach [31]. A multi-objective objective function can be formulated in two ways: a priori and a posteriori approaches [32]. In the a priori approach, objectives are assigned weights according to their relative significance. These objectives are combined to form a fitness function. In contrast, the a posteriori technique identifies a set of non-dominated solutions, providing multiple trade-off options for decision-makers. The priori technique is utilized to design the fitness function, which consists of two key components: TET and TEC. Equation 2.1 presents the fitness function.

$$f(TET, TEC) = \omega_1 \times TET_W + \omega_2 \times TEC_W \tag{2.1}$$

Here, the weight ω_1 is for the objective TET, while the weight ω_2 is for the TEC. The following subsections describe of TET_W and TEC_W of the workflow.

2.1.1. Total Execution Time (Makespan). The makespan denotes the total time taken to execute all tasks assigned across different virtual machines [33]. The makespan of the workflow can be mathematically calculated using Equation 2.2.

$$TET_W = max\{T_i^C | i = 1, 2, ...m\} \tag{2.2}$$

Here, T_i^C represents the completion time of task T_i.

When tasks are interdependent, the completion time also includes the waiting time for preceding tasks to finish. The T_i^C is represented in Equation 2.3.

$$T_i^C = \begin{cases} ET_i & \text{if predce } (T_i) = 0 \\ WT_i + ET_i & \text{if predce } (T_i) \neq 0 \end{cases} \tag{2.3}$$

As shown in Equation 2.4, the waiting time WT_i of task T_i is equal to highest completion time of all its predecessor (predce) tasks.

$$WT_i = \begin{cases} 0 & \text{if predce } (T_i) = 0 \\ Max(T_i^C) & \text{if predce } (T_i) \neq 0 \end{cases} \tag{2.4}$$

$$ET_{ij} = \frac{Size_{Task}}{num(PE_j) \times PE_{unit}} \tag{2.5}$$

Equation 2.5 calculates the execution time. The following symbols $num(PE_j)$, $Size_{Task}$, and PE_{unit} denote the cores of virtual machines, task size, and processing capacity of each core, respectively.

2.1.2. Total Execution Cost. The pay-as-you-go payment model in cloud computing emphasizes cost optimization. The primary components of cloud computing costs are execution, storage, and data transmission. The total execution cost of a virtual machine (VM) comprises both the execution time and the cost per unit time interval. Equation 2.6 defines the total execution cost TEC_W for a workflow [34]. Here, C[i] represents the execution cost per unit time interval for the i^{th} virtual machine. For the purposes of this analysis, C is assumed to be 1 for all virtual machines.

$$TEC_W = \sum_{i=1,}^{VM} C[i] \times (LET[i] - LST) \tag{2.6}$$

$$LET[i] = max(ET[i]) \tag{2.7}$$

$$LST[i] = min(ET[i]) \tag{2.8}$$

Equations 2.7 and 2.8 represent the calculation of least execution time ($LET[i]$) and least start time ($LST[i]$) respectively.

3. Adaptive Weights Grey Wolf Optimization (Proposed Algorithm)

Mirjalili et al. [20] introduced the original GWO. There are four types of leaders in a wolf's pack, such as alpha, beta, delta, and omega, and all leaders participating in hunting play the same role. In our proposed algorithm (AWGWO), the pack of wolves are hierarchically categorized into five leaders: alpha, beta, delta, omega, and gamma. These leaders are participating in hunting with adaptive weights. Categorizing the wolves into five leaders shows enhanced diversity of leaders [35]. In this proposed algorithm, the nature of the weights is adaptive instead of average weights like the original GWO. The proposed algorithm demonstrates that it achieves well exploration in the early phase and exploitation in the latter phase. So this algorithm maintains a good balance between exploration and exploitation. The algorithm demonstrates the following steps.

3.1. Encircling:

In GWO algorithm, grey wolves simulate the encircling of prey during the hunting process. The relevant equations are as follows:

$$\zeta = |\kappa.X_P(t) - X(t)| \tag{3.1}$$

$$X(t+1) = X_P(t) - \mu.\zeta \tag{3.2}$$

Equations 3.1 and 3.2 update the position of the wolves at each iteration t. Here, X_p denotes the position of the prey, and X refers to the current position of the wolf. The coefficients μ and κ are determined using Equations 3.3 and 3.4, respectively.

$$\mu = 2.a.\chi_1 - a \tag{3.3}$$

$$\kappa = 2.\chi_2 \tag{3.4}$$

The random variables χ_1 and χ_2 each take values in the range from 0 to 1. In contrast, Equation 3.8 describes the linear decreases of a from 2 to 0.

3.2. Hunting:

The social hierarchy in grey wolf optimization designates the alpha wolf (α) as the primary leader, directing the hunt, while the beta (β), delta (δ), and gamma (γ) wolves support the alpha with decision-making. The omega (ω) wolves follow the leadership of higher-ranking wolves. In a hypothetical search space, we do not know exactly where the prey is located. In mathematics, the wolves are categorised as α, β, δ, γ, and ω. Only the top four leaders have the information about the location of the prey. Other search agents adjust their positions by following four leaders to exploit their solutions.

$$\zeta_\alpha = |\kappa_1.X_\alpha - X|,\ \zeta_\beta = |\kappa_2.X_\beta - X|,\ \zeta_\delta = |\kappa_3.X_\delta - X|,\ \zeta_\gamma = |\kappa_4.X_\gamma - X| \tag{3.5}$$

$$X_1 = X_\alpha - \mu_1.\zeta_\alpha,\ X_2 = X_\beta - \mu_2.\zeta_\beta,\ X_3 = X_\delta - \mu_3.\zeta_\delta,\ X_4 = X_\gamma - \mu_4.\zeta_\gamma \tag{3.6}$$

$$X(t+1) = \frac{X_1 + X_2 + X_3 + X_4}{4} \tag{3.7}$$

3.3. Attacking:

The grey wolf chases its prey until it stops moving. In math, the value of a gets smaller each time we repeat the process. Equation 3.8 denotes the controlling variable a. In this equation, ι stands for the recent iteration, and τ stands for the maximum iteration.

$$a = 2 \times (1 - \frac{\iota}{\tau}) \tag{3.8}$$

3.4. Position Update with Adaptive Weights

From the Equation 3.7, all dominant wolves participate in the search for the prey with equal weight. Each grey wolf moves towards or away from the leaders with equal influence of the alpha, beta, delta, and gamma wolves. The alpha wolf is closest to the prey at the start of the search. So, we should focus more on the alpha wolf by giving it higher weight than the beta, delta, and gamma wolves [37]. However, it is still far from the best solution. This problem is even bigger for the beta, delta, and gamma wolves. The equal weights for all the dominants are beyond the hierarchical structure of the grey wolves' pack. The alpha wolf is usually the closest to the prey and should have more importance than the others. Consequently, the weight of the beta is always equal to the delta and gamma. The following hypothesis is proposed.

1. In the search and hunting process, the alpha primarily handles the movement, the beta plays a secondary role, the delta has a lesser influence, and the gamma has the least influence among the leaders. All other grey wolves update their positions toward the alpha if it holds the best solution.

2. In real search and hunting, the best position refers to being closest to the prey. In contrast, in optimization problems, the best position corresponds to the maximum or minimum fitness value under the given constraints.

3. In this method, dominant wolves surround the target during search, with α closest, followed by β , δ in third rank, and γ in order of rank, while the remaining ω wolves support the hunt.

At the start of the search, the alpha wolf is usually closest to the prey, so its position is used most to guide the others. In this stage, the weight of alpha is set close to 1.0, while the beta, delta, and gamma have very small weights. As iterations continue, the beta, delta, and gamma start to play a bigger role as they hold the second, third, and fourth best positions. Near the end of the search, when the wolves close in on the prey, all four leaders, alpha, beta, delta, and gamma, are given similar weights denoted in Equation 3.7. In searching from the beginning to the end, the beta follows the alpha since it holds the second position, the delta follows the beta with third rank, and the gamma follows the delta with fourth rank. This means the weights of the beta, delta, and gamma wolves increase as the number of iterations goes up. So, the weight of the alpha should be reduced, and the weights of beta, delta, and gamma are arise.

The above concept can be expressed mathematically. The weights should vary during the search process, but their sum must remain equal to 1.0. Accordingly, Equation 3.7 is modified as follows:

$$X(t+1) = w_1 X_1' + w_2 X_2' + w_3 X_3' + w_4 X_4' \tag{3.9}$$

where X_1', X_2', X_3', X_4' are the updated positions relative to each leader after applying the encircling and hunting rules and the sum of all weights is normalized to:

$$w_1 + w_2 + w_3 + w_4 = 1 \tag{3.10}$$

The weights of the alpha w_1, beta w_2, delta w_3, and gamma w_4 should satisfy the condition $w_1 \geq w_2 \geq w_3 \geq w_4$ at all times. During the search process, the weight of alpha decreases gradually from 1.0 to $\frac{1}{4}$, while the weights of the beta, delta, and gamma increase from 0.0 to $\frac{1}{4}$. A cosine function can be used to model w_1, with the angle θ restricted to vary within $[0, \arccos(\frac{1}{4})]$.

The weights should change according to the increasing number of iterations (it). At the start of the search ($it \to 0$), w_2, w_3, $w_4 \to 0$. As the number of iterations approaches infinity ($it \to \infty$), the weights converge to $w_1, w_2, w_3, w_4 \to \frac{1}{4}$. To model this behavior, an arctangent function of it is introduced, varying from 0.0 to $\frac{\pi}{2}$. Since $\sin\left(\frac{\pi}{4}\right) = \cos\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2}$, an additional angular parameter ϕ is defined in Equation 3.11

$$\phi = \frac{1}{2}\arctan(it) \tag{3.11}$$

Since w_2 increases from 0.0 to $\frac{1}{4}$ as it increases, we assume that it can be expressed in terms of $\sin\theta$ and $\cos\phi$, where $\theta \to \arccos\left(\frac{1}{4}\right)$ as $it \to \infty$. The θ is represented in the Equation 3.12.

$$\theta = \frac{2}{\pi}\arccos\left(\frac{1}{4}\right)\arctan(it) \tag{3.12}$$

Since w_3 increases from 0.0 to $\frac{1}{4}$ as it increases, we assume that it can be expressed in terms of $\sin\theta$ and $\sin\phi$. As $it \to \infty$ and $\phi \to \frac{\pi}{2}$, ensuring that $w_3 \to \frac{1}{4}$ in the later stages of the optimization. We can formulate w_1, w_2, w_3, and w_4 in detail. In the above observations, a new position update mechanism using adaptive weights is presented in the following section.

$$w_1 = \cos(\theta) \tag{3.13}$$

$$w_2 = \frac{1}{2.735}\sin(\theta)\cos(\phi) \tag{3.14}$$

$$w_3 = \frac{1}{2.799}\sin(\theta)\sin(\phi) \tag{3.15}$$

$$w_4 = 1 - w_1 - w_2 - w_3 \tag{3.16}$$

As illustrated in Figure 3, the weight of the α wolf is initially dominant and gradually decreases with the number of iterations, while the weights of the β, δ, and γ wolves increase progressively. In the later iterations, all four weights converge to almost equal values, enabling cooperative exploitation and stable convergence of the algorithm.

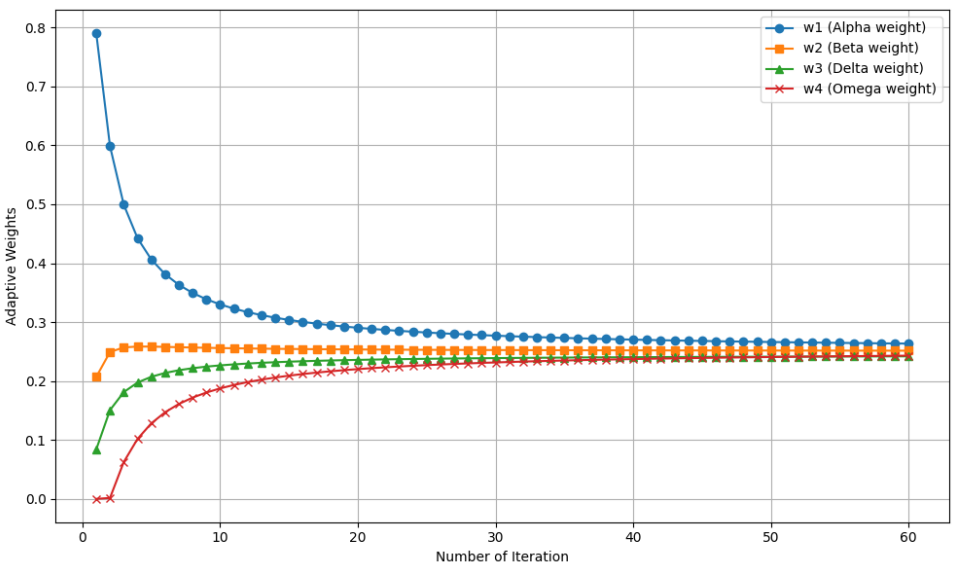

Figure 3: The adaptive weight versus the number of iterations

3.5. Declined Exponentially of Control Parameter

In Equation 3.8, the control parameter decreases linearly, but in practice, faster early reduction improves convergence while maintaining some global search to avoid local optima. Hence, an exponential decay is adopted to balance exploration at the start and exploitation near convergence.

$$a = a_{\max}e^{-\frac{it}{M}} \tag{3.17}$$

Here, a_{max} denotes the initial maximum value and M represents the maximum iteration, while it represents the current iteration, which is selected sufficiently large to prevent excessive runtime or non-convergence. In our experimental result, for workflow scheduling, we choose M is 500. The Algorithm 1 and Figure 4 explain the working of the proposed algorithm in workflow scheduling.

Algorithm 1 AWGWO workflow scheduling algorithm

Input: Workflow tasks $T = \{t_1, t_2, \ldots, t_n\}$, Virtual Machines $VM = \{vm_1, vm_2, \ldots, vm_m\}$, maximum iterations $it_{\max}$, population size N
Output: Optimized task-to-VM assignment minimizing makespan and cost
Initialization: Initialize the positions of N wolves (candidate solutions) randomly in the search space (task–VM mappings)
Evaluate fitness of each wolf using workflow scheduling objective function
Identify α, β, δ, and γ wolves (best four solutions)
for $it = 1$ to $it_{\max}$ **do**
    Update control parameter:
    $a = a_{\max} e^{-\frac{it}{M}}$
    Compute dynamic weights:
    $w_1 = \cos(\theta), \quad w_2 = \frac{1}{2.735}\sin(\theta)\cos(\phi)$
    $w_3 = \frac{1}{2.799}\sin(\theta)\sin(\phi), \quad w_4 = 1 - w_1 - w_2 - w_3$
    where: $\theta = \frac{2}{\pi}\arccos\left(\frac{1}{4}\right)\arctan(it)$, $\phi = \frac{1}{2}\arctan(it)$,
    for each wolf X in population **do**
        Compute distance to each leader: $\zeta = |\kappa.X_P(t) - X(t)|, \quad \kappa = 2.\chi_2$
        Update position relative to each leader: $X(t) = X_P(t) - \mu.\zeta, \quad \mu = 2.a.\chi_1 - a$
        Update wolf's new position using weighted leaders: $X(t+1) = w_1 X'_\alpha + w_2 X'_\beta + w_3 X'_\delta + w_4 X'_\gamma$
        Repair solution to maintain valid task–VM mapping
        Evaluate fitness of updated solution
    end for
    Update α, β, δ, and γ wolves based on new fitness values
end for
return α wolf (best task–VM mapping)

Table 2: Parameters of proposed algorithm

Parameter	**Description**	**Value**
a_{max}	Initial maximum control parameter	2.0
M	Maximum iterations	500
N	Population size (wolves)	25
w_1	Initial weight of α leader	1.0 (decreases to 0.25)
w_2	Initial weight of β leader	0.0 (increases to 0.25)
w_3	Initial weight of δ leader	0.0 (increases to 0.25)
w_4	Initial weight of γ leader	0.0 (increases to 0.25)
VM	Number of virtual machines	Problem-specific
T	Number of tasks	Problem-specific

Table 3: Parameters of Data-center

Parameter	**Value**	**Parameter**	**Value**
No of task (T)	20 to 1000	Bandwidth	1000
VM policy	Time shared	No of data-center	1
No. of processor (P)	1		

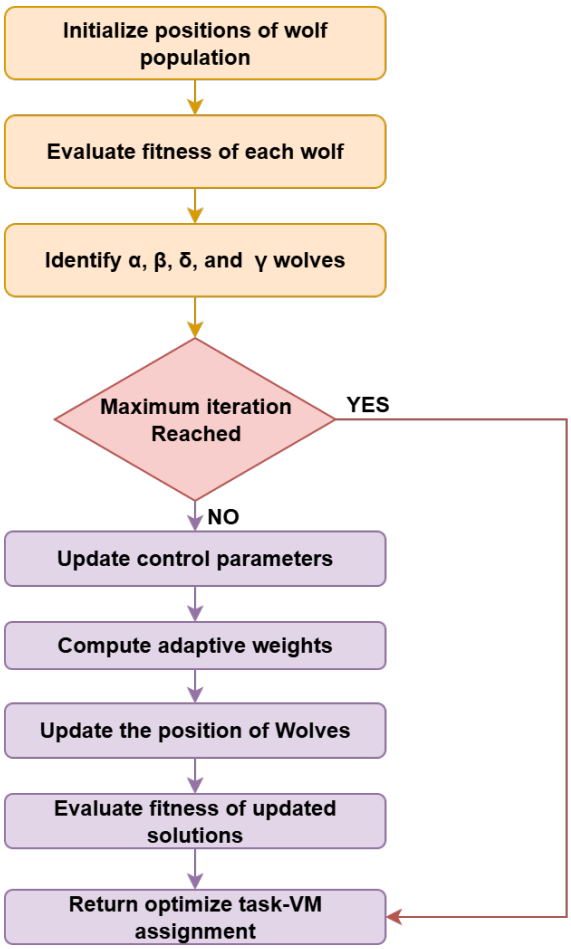

Figure 4: Flowchart of proposed algorithm

4. Result Analysis

The proposed (AWGWO) algorithm was tested under four different scientific workflow scenarios to measure its performance using two fitness functions: TET and TEC. The workflows considered for evaluation include CyberShake, Montage, Inspiral, SIPHT, and Epigenomics with multiple task sizes. For example, Inspiral includes task sets of 30, 50, 100, and 1000 tasks. The Table 4 shows the study of scientific workflows with key characteristics and applications. All practicals were executed on a system configured with an Intel(R) Core i5-5200U processor running at 2.2 GHz, a Windows 8 Pro (64-bit) operating system, and 4 GB of RAM. The WorkflowSim-1.1 toolkit, which extends the CloudSim simulator, was employed to model and analyze the scheduling performance of the proposed approach. The Table 5 and Table 6 represent the TEC and TET of considered workflows, while Table 7 denotes the fitness values of fitness function.

Figure 5 (a) shows the comparison of TEC for the CyberShake workflow with different considered tasks using ACO, GWO, and the proposed AWGWO algorithm. The results clearly show that the AWGWO minimizes TEC significantly compared to ACO and provides noticeable improvement over GWO. The AWGWO reduces TEC by 32.79%, 60.44%, 43.79%, and 39.94% compared to ACO for 30, 50, 100, and 1000 tasks, respectively. Similarly, AWGWO shows 7.28%, 0.47%, 3.63%, and 0.88% reduction over GWO for the same task sizes. Figure 5 (b) illustrates the CyberShake scientific workflow with the following different task sizes (30, 50, 100, and 1000) using ACO, GWO, and the proposed AWGWO algorithm in terms of TET. The results show that AWGWO significantly reduces TET compared to ACO and provides a reduction over GWO. The AWGWO algorithm reduces TET by 34.64%, 59.31%, 44.03%, and 40.35%

Table 4: Study of scientific workflows

Workflow	Key Characteristics	Applications
Epigenomics [26]	Analyzes large-scale DNA sequences, demanding high disk throughput for reading and writing data.	Epigenetic data analysis, genome annotation.
Sipht [24]	Searches for sRNA encoding genes by processing large genomic databases, with heavy data reads and writes.	Bioinformatics, genomic data mining.
Inspiral [27]	Analyzes gravitational wave data, needing extensive computation for waveform matching and filtering.	Gravitational wave detection (LIGO, Virgo).
Montage [23]	Constructs astronomical image mosaics, requiring large data input/output for image processing.	Astrophotography, sky survey mosaics.
Cybershake [25]	Generates earthquake simulations, requiring complex seismic wave propagation calculations.	Earthquake modeling, hazard map generation.

compared to ACO for the considered tasks, respectively. When compared to GWO, AWGWO achieves 4.08%, 4.74%, 1.58%, and 1.47% reduction in TET for the same task sizes. These outcomes demonstrate that AWGWO reduces TET compared to ACO and achieves a moderate reduction over GWO, making it an efficient approach for workflow scheduling in cloud environments.

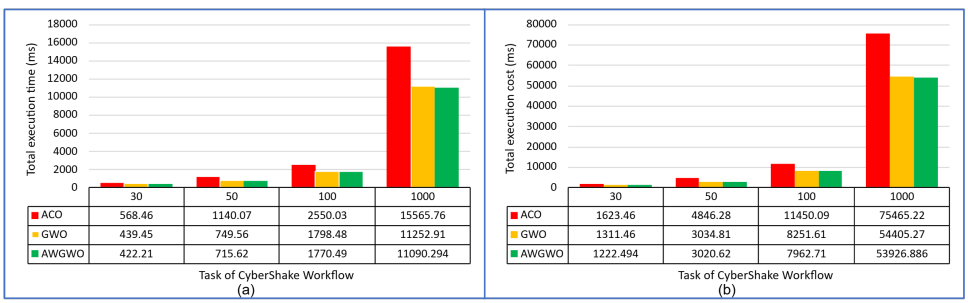

Figure 5: (a) TEC and (b) TET in CyberShake scientific workflow

Figure 6 (a) presents the TEC comparison for the Inspiral workflow with varying task sizes using ACO, GWO, and the proposed AWGWO algorithm. The results indicate that AWGWO achieves significant improvement over ACO and a slight improvement over GWO. The AWGWO algorithm reduces TEC by 6.49%, 1.69%, 19.54%, and 5.13% compared to ACO for 30, 50, 100, and 1000 tasks, respectively. When compared to GWO, the improvements are 1.58%, 0.49%, 0.10%, and 0.74% for the respective task sizes. Figure 6 (b) illustrates the TET comparison for the Inspiral scientific workflow using ACO, GWO, and the proposed AWGWO algorithm for 30, 50, 100, and 1000 tasks. The AWGWO algorithm decreases in TET by 3.96%, 40.32%, and 24.61% over ACO for 50, 100, and 1000 tasks, respectively, while an increase of 12.75% is observed for 30 tasks. The AWGWO consistently reduces TET with 11.16%, 10.80%, 3.96%, and 1.24% for 30, 50, 100, and 1000 tasks, respectively, compared to GWO.

Figure 7 (a) denotes the TEC under the Montage scientific workflow using ACO, GWO, and the proposed AWGWO algorithm. The proposed AWGWO algorithm achieves a consistent reduction over both ACO and GWO. The AWGWO reduces TEC by 1.97%, 0.94%, 0.08%, and 1.29% compared to ACO for 25, 50, 100, and 1000 tasks, respectively. The proposed algorithm reduces by 0.78%, 0.85%, 0.60%, and 1.15% compared to GWO for the same task sizes. Figure 7 (b) illustrates the Total Execution Time (TET) comparison for the Montage scientific workflow across 25, 50, 100, and 1000 tasks using ACO, GWO, and the proposed AWGWO algorithm. The results show that AWGWO offers competitive performance compared to the other algorithms. The proposed algorithm compared to ACO, TET reduces

Table 5: Total execution cost with different workflows

					Percentage Reduction	
Workflow	**Tasks**	**ACO**	**GWO**	**AWGWO**	**From ACO**	**From GWO**
CyberShake	30	1623.46	1311.46	**1222.49**	32.81	7.27
	50	4846.28	3034.81	**3020.62**	60.44	0.46
	100	11450.09	8251.61	**7962.71**	43.80	3.62
	1000	75465.22	54405.27	**53926.89**	39.93	0.88
Inspiral	30	8554.24	8159.96	**8033.18**	6.48	1.57
	50	14231.32	14063.63	**13995.24**	1.69	0.48
	100	28459.09	23830.97	**23807.16**	19.54	0.10
	1000	252813.3	242255.5	**240479.6**	5.12	0.74
Montage	25	293.96	290.53	**288.26**	1.97	0.79
	50	631.14	630.56	**625.21**	0.95	0.85
	100	1304.27	1310.98	**1303.12**	0.09	0.60
	1000	12772.99	12755.65	**12609.5**	1.29	1.15
SIPHT	30	20448.32	11380.29	**11215.42**	82.32	1.47
	60	31194.71	24784.39	**22333.65**	39.67	10.97
	100	45751.31	38485.94	**36393.47**	25.71	5.74
	1000	592117.6	494838.8	**490336.9**	20.76	0.91
Epigenomics	24	21593.16	18703.96	**18026.1**	19.79	3.76
	47	47942.12	**44900.29**	46506.67	3.09	-3.45
	100	422283.1	414633.6	**408862.1**	3.28	1.41
	997	4874893	3928125	**3926024**	24.17	0.05

Table 6: Total execution time with different workflows

					Percentage Reduction	
Workflow	**Tasks**	**ACO**	**GWO**	**AWGWO**	**From ACO**	**From GWO**
CyberShake	30	568.46	439.45	**422.21**	34.64	4.08
	50	1140.07	749.56	**715.62**	59.31	4.74
	100	2550.03	1798.48	**1770.49**	44.02	1.58
	1000	15565.76	11252.91	**11090.29**	40.35	1.46
Inspiral	30	**1974.45**	2515.54	2263.04	-12.75	11.15
	50	3627.57	3866.08	**3489.25**	3.96	10.79
	100	7647.39	5665.73	**5450.14**	40.32	3.96
	1000	61241.08	49755.23	**49144.48**	24.61	1.25
Montage	25	**67.23**	84.79	86.93	-22.61	-2.46
	50	**136.45**	153.49	148.80	-8.29	3.15
	100	**278.44**	290.79	285.62	-2.51	1.81
	1000	**2640.57**	2683.97	2678.49	-1.41	0.20
SIPHT	30	5400.30	4424.60	**4416.78**	22.27	0.18
	60	8338.89	6820.25	**6582.59**	26.68	3.61
	100	11009.13	9039.24	**8902.89**	23.65	1.53
	1000	120032.7	100337.1	**98578.96**	21.76	1.78
Epigenomics	24	**6717.80**	7846.05	8078.43	-16.84	-2.87
	47	**13140.41**	13442.22	13398.42	-1.92	0.32
	100	**95277.34**	121109.3	111411.4	-14.48	8.70
	997	1201141	876848.8	**854301.4**	40.59	2.63

Table 7: Fitness

Workflow	Tasks	ACO	GWO	AWGWO
CyberShake	30	1095.96	875.455	**822.352**
	50	2993.175	1892.185	**1868.12**
	100	7000.06	5025.045	**4866.60**
	1000	45515.49	32829.09	**32508.59**
Inspiral	30	5264.345	5337.75	**5148.111**
	50	8929.445	8964.855	**8742.243**
	100	18053.24	14748.35	**14628.65**
	1000	157027.2	146005.4	**144812.0**
Montage	25	**180.595**	187.66	187.595
	50	**383.795**	392.025	387.005
	100	**791.355**	800.885	794.37
	1000	7706.78	7719.809	**7643.991**
SIPHT	30	12924.31	7902.445	**7816.10**
	60	19766.80	15802.32	**14458.12**
	100	28380.22	23762.59	**22648.18**
	1000	356075.2	297587.9	**294457.9**
Epigenomics	24	14155.48	13275.01	**13052.27**
	47	30541.27	**29171.26**	29952.55
	100	258780.2	267871.4	**260136.7**
	997	3038017	2402487	**2390163**

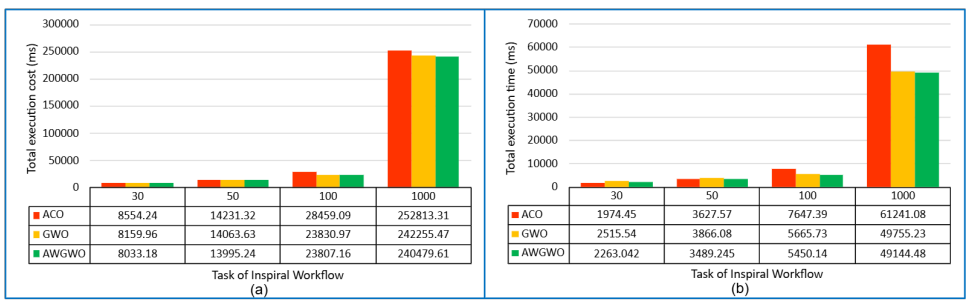

Figure 6: (a) TEC and (b) TET in Inspiral scientific workflow

by -22.66%, -8.29%, -2.51%, and -1.41% for 25, 50, 100, and 1000 tasks, respectively. Against GWO, the reduces are -2.46%, 3.15%, 1.81%, and 0.20% for the same task sizes.

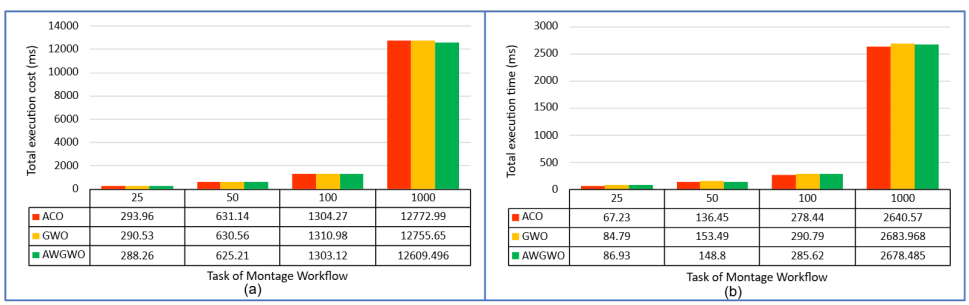

Figure 7: (a) TEC and (b) TET in Montage scientific workflow

Figure 8 (a) illustrates the TEC under the SIPHT scientific workflow using ACO, GWO, and the

proposed AWGWO algorithm. The AWGWO algorithm significantly reduces TEC compared to ACO and GWO. The Proposed AWGWO algorithm reduces TEC by 82.32%, 39.67%, 25.71%, and 20.75% compared to ACO for 30, 60, 100, and 1000 tasks, respectively. When compared to GWO, the reductions are 1.47%, 10.97%, 5.74%, and 0.91% for the same task sizes. Figure 8 (b) illustrates the TET under the SIPHT scientific workflow using ACO, GWO, and the proposed AWGWO algorithm. The AWGWO algorithm reduces TET when compared to ACO and GWO. Specifically, AWGWO reduces TET by 22.26%, 26.68%, 23.65%, and 21.76% over ACO for 30, 60, 100, and 1000 tasks, respectively. In comparison to GWO, the reductions are 0.17%, 3.61%, 1.53%, and 1.78% for the same task sizes.

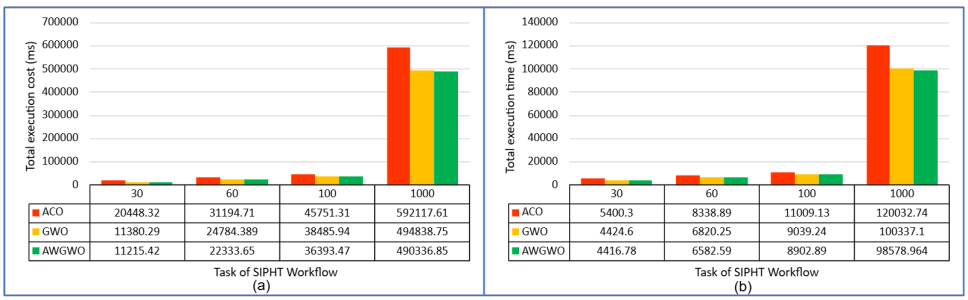

Figure 8: (a) TEC and (b) TET in SIPHT scientific workflow

Figure 9 (a) represents the TEC under the Epigenomics workflow using ACO, GWO, and the proposed AWGWO algorithm. The AWGWO algorithm demonstrates that the reductions of TEC by 19.78%, 3.08%, 3.28%, and 24.16% for 24, 47, 100, and 997 tasks, respectively, compared to ACO. The AWGWO shows a reduction of 3.76% for 24 tasks, while for 47 tasks, a slight increase of -3.45% is observed. The proposed algorithm reduces TEC for 100 and 997 tasks; the reductions are 1.41% and 0.05%, respectively. Figure 9 (b) illustrates the TET under the Epigenomics workflow using ACO, GWO, and the proposed AWGWO algorithm. The proposed algorithm reduces the TET by -16.84%, -1.92%, and -14.48% for 24, 47, and 100 tasks, respectively, but achieves a significant improvement of 40.59% for 997 tasks compared to ACO. The AWGWO algorithm reduces by -2.87%, 0.32%, and 8.70% for 24, 47, and 100 tasks, and 2.63% for 997 tasks compared to GWO.

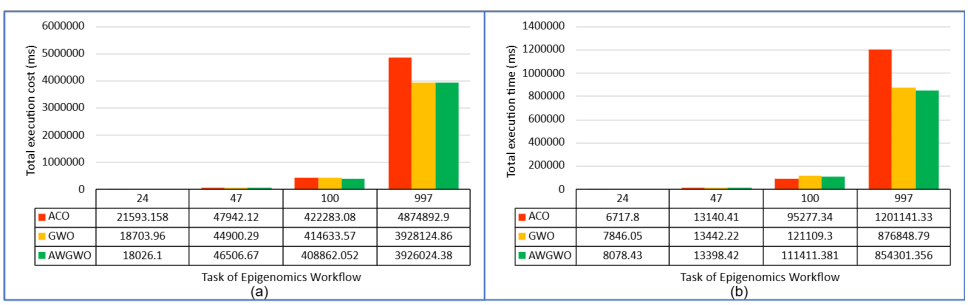

Figure 9: (a) TEC and (b) TET in Epigenomics scientific

5. Conclusion

The proposed AWGWO improves TET and TEC by adaptively balancing exploration and exploitation through its two-phase search mechanism. The early phase is useful for enhancing the exploration search area, while the latter phase is useful for exploitation. Compared to ACO and GWO, AWGWO provides better scheduling performance by effectively balancing exploration and exploitation, resulting in lower TET and TEC for both small and large scientific workflows. So, this approach provides a better trade-off between exploration and exploitation, which results in significant improvements in both TEC and TET. For instance, under the CyberShake workflow, AWGWO reduced up to 60.43% reduction in TEC

compared to ACO and 7.27% compared to GWO for 50 and 30 tasks, respectively. Similarly, in the SIPHT workflow, TEC was reduced by 82.32% over ACO and 10.97% over GWO for 30 and 60 tasks. In the case of the Inspiral workflow, AWGWO lowered TEC by 19.54% over ACO for 100 tasks and 1.57% over GWO for 30 tasks. The AWGWO reduced by 1.97% compared to ACO and 0.78% compared to GWO under the montage scientific workflow with 25 tasks. In Epigenomics, AWGWO showed strong results for large tasks, with 24.16% TEC reduction compared to ACO for 997 tasks. In terms of TET, AWGWO also provided notable improvements, such as a 44.02% reduction compared to ACO in CyberShake with 100 tasks and a 40.31% reduction in Inspiral for 100 tasks.

The computational complexity of the proposed AWGWO algorithm is primarily $O(N \times T \times D)$, where N is the population size, T is the iterations, and D is the number of tasks in the workflow. However, a few limitations were observed, such as minor performance variations for smaller workflows and additional computational effort due to adaptive weight adjustments. This study considers only static scientific workflows, assuming predefined task structures and resource information, and does not address dynamic scheduling workflows. Moreover, as the number of virtual machines increases, execution time reduces due to better parallelism, but execution cost increases because of higher resource usage. For the future, AWGWO can be enhanced by integrating with deep learning models or hybrid metaheuristics.

Acknowledgments

The authors express sincere gratitude to Rajasthan Technical University Kota for invaluable support and resources throughout the research.

Ethics Approval and Consent to Participate

Not applicable.

Consent for Publication

Not applicable.

Availability of Data and Material

The data will be available based on readers request.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Mehta, S. and Kaur, P., *Scheduling Data Intensive Scientific Workflows in Cloud Environment Using Nature Inspired Algorithms*, in *Nature-Inspired Algorithms for Big Data Frameworks*, IGI Global Scientific Publishing, pp. 196–217, (2019).

2. Fernández-Cerero, D., Jakóbik, A., Fernández-Montes, A., and Kołodziej, J., *GAME-SCORE: Game-based Energy-Aware Cloud Scheduler and Simulator for Computational Clouds*, Simulation Modelling Practice and Theory, Vol. 93, pp. 3–20, Elsevier, (2019).

3. Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., and Wilkes, J., *Omega: Flexible, Scalable Schedulers for Large Compute Clusters*, in *Proceedings of the 8th ACM European Conference on Computer Systems*, pp. 351–364, (2013).

4. Fernández-Cerero, D., Ortega-Irizo, F. J., Fernández-Montes, A., and Velasco-Morente, F., *Bullfighting Extreme Scenarios in Efficient Hyper-Scale Cluster Computing*, Cluster Computing, Vol. 23, No. 4, pp. 3387–3403, Springer, (2020).

5. Hammad, R., Barhoush, M., and Abed-Alguni, B. H., *A Semantic-Based Approach for Managing Healthcare Big Data: A Survey*, Journal of Healthcare Engineering, Vol. 2020, No. 1, Article ID 8865808, Wiley Online Library, (2020).

6. Dinh, H. T., Lee, C., Niyato, D., and Wang, P., *A Survey of Mobile Cloud Computing: Architecture, Applications, and Approaches*, Wireless Communications and Mobile Computing, Vol. 13, No. 18, pp. 1587–1611, Wiley Online Library, (2013).

7. Rajagopalan, A., Modale, D. R., and Senthilkumar, R., *Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm*, in *Proceedings of the International Conference on E-Business and Telecommunications*, pp. 678–687, Springer, (2019).

8. Alzaqebah, A., Al-Sayyed, R., and Masadeh, R., *Task Scheduling Based on Modified Grey Wolf Optimizer in Cloud Computing Environment*, in *Proceedings of the 2019 2nd International Conference on New Trends in Computing Sciences (ICTCS)*, pp. 1–6, IEEE, (2019).

9. Ghafarian, T., Javadi, B., and Buyya, R., *Decentralised Workflow Scheduling in Volunteer Computing Systems*, International Journal of Parallel, Emergent and Distributed Systems, Vol. 30, No. 5, pp. 343–365, Taylor & Francis, (2015).

10. Kumar, S., and Buyya, R., *Green Cloud Computing and Environmental Sustainability*, in *Harnessing Green IT: Principles and Practices*, pp. 315–339, Wiley Online Library, (2012).

11. Patel, G., Mehta, R., and Bhoi, U., *Enhanced Load Balanced Min-Min Algorithm for Static Meta Task Scheduling in Cloud Computing*, Procedia Computer Science, Vol. 57, pp. 545–553, Elsevier, (2015).

12. Mao, Y., Chen, X., and Li, X., *Max–Min Task Scheduling Algorithm for Load Balance in Cloud Computing*, in *Proceedings of International Conference on Computer Science and Information Technology*, S. Patnaik and X. Li (Eds.), pp. 457–465, Springer India, New Delhi, (2014).

13. Ilavarasan, E., and Thambidurai, P., *Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments*, Journal of Computer Sciences, Vol. 3, No. 2, pp. 94–103, Citeseer, (2007).

14. S. Pandey, L. Wu, S. M. Guru, and R. Buyya, *A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments*, *Proc. 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pp. 400–407 (2010), IEEE.

15. Awad, A. I., El-Hefnawy, N. A., and Abdel-Kader, H. M., *Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments*, Procedia Computer Science, Vol. 65, pp. 920–929, Elsevier, (2015).

16. Khalili, A., and Babamir, S. M., *Optimal Scheduling Workflows in Cloud Computing Environment Using Pareto-Based Grey Wolf Optimizer*, Concurrency and Computation: Practice and Experience, Vol. 29, No. 11, Article e4044, Wiley Online Library, (2017).

17. Chen, Z.-G., Zhan, Z.-H., Lin, Y., Gong, Y.-J., Gu, T.-L., Zhao, F., Yuan, H.-Q., Chen, X., Li, Q., and Zhang, J., *Multiobjective Cloud Workflow Scheduling: A Multiple Populations Ant Colony System Approach*, IEEE Transactions on Cybernetics, Vol. 49, No. 8, pp. 2912–2926, IEEE, (2018).

18. Choudhary, A., Gupta, I., Singh, V., and Jana, P. K., *A GSA Based Hybrid Algorithm for Bi-Objective Workflow Scheduling in Cloud Computing*, Future Generation Computer Systems, Vol. 83, pp. 14–26, Elsevier, (2018).

19. Liu, L., Zhang, M., Buyya, R., and Fan, Q., *Deadline-Constrained Coevolutionary Genetic Algorithm for Scientific Workflow Scheduling in Cloud Computing*, Concurrency and Computation: Practice and Experience, Vol. 29, No. 5, Article e3942, Wiley Online Library, (2017).

20. Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis, *Grey wolf optimizer*, *Advances in Engineering Software*, vol. 69, pp. 46–61, Elsevier (2014).

21. Bansal, N., and Singh, A. K., *Grey Wolf Optimized Task Scheduling Algorithm in Cloud Computing*, in *Frontiers in Intelligent Computing: Theory and Applications: Proceedings of the 7th International Conference on FICTA (2018), Volume 1*, pp. 137–145, Springer, (2019).

22. Wang, H., Zhang, J., Fan, J., Zhang, C., Deng, B., and Zhao, W., *An Improved Grey Wolf Optimizer with Flexible Crossover and Mutation for Cluster Task Scheduling*, Information Sciences, Vol. 704, Article 121943, Elsevier, (2025).

23. Jiang, Q., Lee, Y. C., Arenaz, M., Leslie, L. M., and Zomaya, A. Y., *Optimizing Scientific Workflows in the Cloud: A Montage Example*, in *2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pp. 517–522, IEEE, (2014).

24. Livny, J., Teonadi, H., Livny, M., and Waldor, M. K., *High-Throughput, Kingdom-Wide Prediction and Annotation of Bacterial Non-Coding RNAs*, PloS ONE, Vol. 3, No. 9, Article e3197, Public Library of Science, San Francisco, USA, (2008).

25. Graves, R., Jordan, T. H., Callaghan, S., Deelman, E., Field, E., Juve, G., Kesselman, C., Maechling, P., Mehta, G., Milner, K., et al., *CyberShake: A Physics-Based Seismic Hazard Model for Southern California*, Pure and Applied Geophysics, Vol. 168, pp. 367–381, Springer, (2011).

26. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., and Vahi, K., *Characterizing and profiling scientific workflows*, Future Generation Computer Systems, Vol. 29, No. 3, pp. 682–692, Elsevier, (2013).

27. Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., et al., *Pegasus: A framework for mapping complex scientific workflows onto distributed systems*, Scientific Programming, Vol. 13, No. 3, pp. 219–237, IOS Press, (2005).

28. Kalra, M., and Singh, S., *Multi-Criteria Workflow Scheduling on Clouds under Deadline and Budget Constraints*, Concurrency and Computation: Practice and Experience, Vol. 31, No. 17, Article e5193, Wiley Online Library, (2019).

29. Loscalzo, J., and Handy, D. E., *Epigenetic Modifications: Basic Mechanisms and Role in Cardiovascular Disease (2013 Grover Conference Series)*, Pulmonary Circulation, Vol. 4, No. 2, pp. 169–174, SAGE Publications, London, England, (2014).

30. Weiwei Chen and Ewa Deelman, *Workflowsim: A toolkit for simulating scientific workflows in distributed environments*, in *Proceedings of the 2012 IEEE 8th International Conference on E-Science*, pp. 1–8, IEEE (2012).

31. Wu, F., Wu, Q., and Tan, Y., *Workflow Scheduling in Cloud: A Survey*, The Journal of Supercomputing, Vol. 71, pp. 3373–3418, Springer, (2015).

32. Mirjalili, S., Saremi, S., Mirjalili, S. M., and Coelho, L. dos S., *Multi-Objective Grey Wolf Optimizer: A Novel Algorithm for Multi-Criterion Optimization*, Expert Systems with Applications, Vol. 47, pp. 106–119, Elsevier, (2016).

33. Ebadifard, F., and Babamir, S. M., *A PSO-Based Task Scheduling Algorithm Improved Using a Load-Balancing Technique for the Cloud Computing Environment*, Concurrency and Computation: Practice and Experience, Vol. 30, No. 12, Article e4368, Wiley Online Library, (2018).

34. Adhikari, M., Amgoth, T., and Srirama, S. N., *A Survey on Scheduling Strategies for Workflows in Cloud Environment and Emerging Trends*, ACM Computing Surveys (CSUR), Vol. 52, No. 4, pp. 1–36, ACM, New York, NY, USA, (2019).

35. Singh, N., *A Modified Variant of Grey Wolf Optimizer*, Scientia Iranica, Vol. 27, No. 3, pp. 1450–1466, Sharif University of Technology, (2020).

36. Lal, C., and Sharma, H., *A Workflow Scheduling Using an Efficient Hybrid PSO-MGWO Algorithm in Cloud Computing*, SN Computer Science, Vol. 6, No. 8, Article 929, Springer, (2025).

37. Gao, Z.-M., and Zhao, J., *An Improved Grey Wolf Optimization Algorithm with Variable Weights*, Computational Intelligence and Neuroscience, Vol. 2019, No. 1, Article 2981282, Wiley Online Library, (2019).

38. Lal, C., Sharma, H., and Sharma, C., *Workflow Scheduling Using MGWO Algorithm in Cloud Computing*, Proceedings of the International Conference on Communication and Intelligent Systems, pp. 17–28, Springer, (2024).

39. Bansal, S., Singla, B. S., and Aggarwal, H., *Cost-Efficient Task Scheduling in Cloud-Fog Systems using Hybrid Algorithm*, Proceedings of the 3rd International Conference on Advancement in Computation & Computer Technologies (InCACCT), pp. 960–963, IEEE, (2025).

40. Lal, C., Sharma, H., and Arora, N., *PSOEGWO: An Efficient Workflow Scheduling Algorithm for Clouds*, SN Computer Science, Vol. 7, No. 1, Article 88, Springer, (2026).

Chotu Lal,
Department of computer science and engineering,
Rajasthan technical university, kota, rajasthan,
India.
E-mail address: clal.phd22@rtu.ac.in
ORCHID iD: https://orcid.org/0009-0002-9355-4960

and

Harish Sharma,
Department of computer science and engineering,
Rajasthan technical university, kota, rajasthan,
India.
E-mail address: hsharma@rtu.ac.in
Orcid ID: https://orcid.org/0000-0001-6685-3338