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

(3s.) **v. 2026 (44)** : 1–14.
ISSN-0037-8712
doi:10.5269/bspm.79383

Multi-Objective Workflow Scheduling Using DWGWO Algorithm in Cloud Computing

Chotu Lal*, Harish Sharma and Vaishali Maheshwari

ABSTRACT: This paper presents an approach for workflow scheduling, which assumes diverse quality of service needs in the clouds. The main target of the scheduling is to minimize total execution time (TET) and total execution cost (TEC) while scheduling to meet the service level agreement (SLA). Workflow scheduling is classified as an NP-hard problem in cloud environments. Conventional algorithms are unable to tackle the workflow scheduling within polynomial time. This article proposes a multi-objective system for the workflow scheduling problem using a Dynamic Weights Grey Wolf Optimization (DW-GWO) algorithm for cloud computing. This algorithm has a balance between exploration and exploitation capabilities. This algorithm helps to minimize the TET and TEC for tasks that depend on each other. The experimental results indicate that the DW-GWO algorithm outperformed the ACO and GWO algorithms in terms of TET and TEC.

Keywords: Optimization algorithms, metaheuristic algorithms, Dynamic Weights Grey Wolf Optimization, cloud computing, workflow scheduling, nature-inspired algorithms.

Contents

1. Introduction

Cloud computing is a way of using the Internet to process data and provide storage and network services. It gives users a lot of flexibility, letting them choose the services they need, often through a subscription. Nowadays, the cloud computing area is enhanced, offering several benefits to users [1]. These benefits are provided by a service provider using the concept of virtualization, where tasks are mapped and executed on virtual machines (VMs), a process known as task scheduling. The task scheduling in cloud environments remains an NP-hard problem due to their dynamic and heterogeneous nature [2]. Workflow scheduling is a process that maps the tasks and virtual machines when tasks are interdependent. Many researchers have designed numerous algorithms to optimize the makespan and cost of the workflow [3,5,6,4,7]. However, there is still scope to further reduce makespan and cost.

In [4], the suggested algorithm manages balanced and imbalanced workflows. This method also optimizes the data transfer time and communication costs. The DiCSPM [4] is utilized to enhance efficiency and reduce communication costs to GWO [30], BRS [33], and PSO [32]. The Genetic Algorithm

* Corresponding author.
2020 *Mathematics Subject Classification:* 68M20, 90C29.
Submitted October 07, 2025. Published June 05, 2026.

(GA) takes much time to reach optimal solutions [8]. Tawfeek et al. [9] utilized the Ant Colony Optimization (ACO) [13] for task scheduling to decrease makespan and observed that ACO surpassed FCFS [34] and RR [35]. However, the ACO algorithm is complex and requires a significant amount of time to reach optimal results [10]. The paper [11] proposes a Grey Wolf Optimised (GWO) task scheduling algorithm to reduce execution cost in cloud computing using CloudSim. The modified Firefly Algorithm (FA) was designed to achieve better load balancing and reduced execution time in cloud computing using workflowsim [12]. The Enhanced Flower Pollination (EFP) algorithm is used for cloud task scheduling to reduce makespan and shows improved performance compared to PBACO [36], ACO [13], Min-Min [37], and FCFS strategies [14]. The paper uses the Bat Algorithm (BA), a metaheuristic inspired by bat echolocation, to reduce execution cost [15]. This paper uses the Artificial Bee Colony (ABC) method to schedule tasks to improve load balancing and resource utilisation [16]. The Cuckoo Search (CS) algorithm is used in workflow scheduling to improve energy efficiency and reduce SLA violations in cloud computing [17]. The Cat Swarm Optimization (CSO) algorithm demonstrates the workflow scheduling problem [6]. This algorithm is used for enhancing resource utilisation.

The literature review and Table 1 highlight a notable research gap as standard metaheuristic algorithms are not well-suited for addressing cloud scheduling problems [2]. These algorithms often suffer from premature convergence to local optima, an imbalance between exploration and exploitation, high computational overhead, and slow convergence rates. To address these identified research gaps, the following paragraph introduces a new approach specifically designed to overcome these limitations.

This research introduces a metaheuristic approach called the Dynamic Weights Grey Wolf Optimization (DW-GWO) algorithm. In the original GWO, the top three leaders are influenced by equal importance. In the proposed approach (DW-GWO), a dynamic weight is adopted for alpha, which is influenced by weight $w_1 = 1.0$ to 0.5, beta is influenced by $w_2 = 0$ to 0.34, and delta is influenced by $w_3 = 0$ to 0.16, and these weights are assigned to these leaders. In DW-GWO, the alpha wolf is super dominant during the search process and provides stronger guidance toward the best solution. The beta and delta provide supportive contributions to preserve population diversity. Consequently, exploration is emphasized during the early iterations, and exploitation is progressively in the later stages. The DW-GWO aims to improve exploration-exploitation balance, accelerate convergence, and avoid premature trapping in local optima. The proposed (DW-GWO) algorithm is used in workflow scheduling to reduce total execution time (TET) and total execution cost (TEC). The workflow scheduling algorithm was simulated using WorkflowSim. Table 2 shows the list of symbols used in this paper with their corresponding definitions.

2. Mathematical Model of Workflow Scheduling and Proposed Algorithm

In this section, we discuss the mathematical model of the fitness function of workflow scheduling, GWO, and the proposed DW-GWO algorithms.

2.1. Fitness function of workflow scheduling

In this subsection, we discuss the fitness function of workflow scheduling. Here a priori approach is used to build a multi-objective function, known as a fitness function. This fitness function consists of the objective parameters with their weights. Here, TEC and TET are the parameters of the fitness function $f(TET, TEC)$. These parameters are integrated with their weights to build a fitness function for multiobjective workflow scheduling.

$$f(TET, TEC) = \gamma_1 \times TET + \gamma_2 \times TEC \tag{2.1}$$

where the value of γ_1 and γ_2 is 0.5.

2.1.1. Total execution time. The computation time (CMT) of task T_q is calculated using Equation 2.8 and this task is assigned to resource R_j [20].

$$CMT(T_q, R_j) = \frac{Length_q}{PC_j} \tag{2.2}$$

Table 1: Comparative discussion of task scheduling algorithms

References	Approaches	Used Workflows	Key Contribution	Limitations
2018 [18]	BBO and HMBBO	4	It achieves a faster convergence rate compared to the considered algorithms.	The workload distribution over resources and the associated cost are not considered
2021 [19]	MOO algorithm	2	Energy consumption is reduced compared with the considered approaches.	This study considers only two scientific workflows for evaluation.
2021 [20]	PSO + LOA	3	The performance in terms of TEC is better compared to PSO and LOA.	Only discussed a single parameter, which is cost.
2021 [21]	PSO-GWO	4	Execution cost and time are effectively reduced compared to GWO and PSO.	The results do not consider resource scalability for large-scale workflows.
2022 [22]	CHGA algorithm	4	CHGA achieves reduced cost compared to the considered algorithms.	Only focused cost parameter.
2023 [23]	Cost-effective algorithm	3	Reduced cost compared to PSO and GA.	Evaluation was conducted using three scientific workflows.
2023 [24]	MOS approach	0	The makespan was reduced compared to considered methods.	The emphasis is solely on reducing costs and minimizing makespan.
2023 [26]	Population based approach	0	Reduced execution time and cost.	No scientific workflows used.
2023 [25]	Novel energy coefficient based algorithm	1	Improvement in energy consumption and data dependency management.	The focus was solely on energy consumption.
2024 [27]	HEPGA algorithm	0	Better at optimizing tasks and fitness values.	Focused solely on minimizing makespan
2025 [28]	MGWO algorithm	3	Multi-objective workflow scheduling.	Used only three scientific workflow.
2025 [29]	HPWOA	3	Hybrid HPWOA for single-objective workflow scheduling.	Only three scientific workflows used and discussed only cost parameter.
2026 [38]	PSOEGWO	4	Hybrid algorithm for multi-objective workflow scheduling.	Only four scientific workflows used.
...	DW-GWO	5	Reduce execution time and execution cost	This approach is not used for energy consumption and other objectives.

Here, $Length_q$ illustrates the length of task T_q in million instructions (MI), and PC_j is the processing capacity of resource R_j. The TT(T_q) denotes the transferring time for task T_q. The transfer time for task T_q is the longest input transfer time of all input files. The Equation 2.3 illustrated the transferring of the task (T_q).

Table 2: Symbol table

Symbol	Definition	Symbol	Definition
TET	Total execution time	CMT	Computation time
TEC	Total execution cost	PC	Processing capacity
T	Task	R	Resource
MI	Million Instructions	TT	Transferring time
PCT	Processing time	BW	Bandwidth
STT	Start time	FT	Finish time
RET	End rental time	RST	Start rental time

$$TT(T_q) = \max_{T_p \in T_q} \{\frac{SZData_{(T_p,T_q)}}{BW}\} \tag{2.3}$$

Here, the task T_p is a predecessor task of task T_q, $SZData_{(T_p,T_q)}$ illustrates the size of data that was transferred from task T_p to T_q, and BW denotes the available bandwidth. The transfer time is considered zero when T_p and T_q are assigned to the same virtual machine (VM). The processing time (PCT) of task T_q scheduled on resource R_j is expressed in Equation 2.4.

$$PCT(T_q, R_j) = TT(T_q) + CMT(T_q, R_j) \tag{2.4}$$

Equation 2.5 is used to calculate the start time (STT) of task (T_q) assigned to the VM resource R_j.

$$STT(T_q, R_j) = \begin{cases} 0, & \text{if prede } (T_q) = 0 \\ \max_{T_p \in T_q}\{STT(T_p, R_j) + PCT(T_q, R_j)\} & \text{if prede } (T_q) \text{ !=0} \end{cases} \tag{2.5}$$

Equation 2.6 is used to calculate the finish time (FT) of task (T_q).

$$FT(T_q, R_j) = \begin{cases} PCT(T_q, R_j), & \text{if prede } (T_q) = 0 \\ \max_{T_p \in T_q}\{FT(T_p) + PCT(T_p, R_j)\} & \text{if prede } (T_q) \text{ !=0} \end{cases} \tag{2.6}$$

The total execution time of a directed acyclic graph is calculated using Equation 2.7.

$$TET = \max_{T_q \in T}\{FT(T_q, R_j)\} \tag{2.7}$$

2.1.2. Total execution cost. The cloud services are provided by cloud service providers (CSPs) to the user based on a pay-per-use model. These services have a fixed price for transferring a unit of data between two VM nodes [20]. The execution cost (EXC) of a virtual machine can be defined by the Equation 2.8.

$$EXC(R_j) = \lfloor \frac{RET_j - RST_j}{\tau} \rfloor * vc_j \tag{2.8}$$

Where vc_j represents the rental price per unit time for resource R_j, and τ is its corresponding unit time interval. The RET_j and RST_j represent the end rental time and the start rental time of the resource R_j, respectively. $\lfloor\ \rfloor$ denotes the floor function, which rounds a value down to the nearest integer. The TEC of the workflow is denoted by Equation 2.9.

$$TEC = \sum_{i=1,}^{n} EXC(R_j) \tag{2.9}$$

Here, n denotes virtual machines. The objective of this work is to minimize the TET and TEC.

2.2. Grey Wolf Optimization algorithm

Mirjalili et al. [30] presented the GWO approach. In GWO, the wolves stay in hierarchical order, such as alpha, beta, delta, and omega. The following steps illustrate the algorithm.

2.2.1. Encircling prey. This step describes how wolves encircle their prey during hunting.

$$\zeta = |\kappa.Z_P(t) - Z(t)| \tag{2.10}$$

$$Z(t+1) = Z_P(t) - \mu.\zeta \tag{2.11}$$

Equations 2.10 and 2.11 help to find the best places of the wolves according to iteration (t). The Z_p and Z illustrate the position of the prey and wolf, respectively. The values of the coefficients μ and κ are computed using Equations 2.12 and 2.13, respectively.

$$\mu = 2.a.\chi_1 - a \tag{2.12}$$

$$\kappa = 2.\chi_2 \tag{2.13}$$

The random variables χ_1 and χ_2 take values in the interval $[0, 1]$, whereas Equation 2.17 defines the linear decrease of the parameter a from 2 to 0.

2.2.2. Hunting. In the hunting process, alpha plays the most critical role by guiding beta and delta, and encircling the prey. The alpha, beta, and delta find the exact location of the prey (optimal solution). The remaining wolves update their location by referencing the leaders to enhance solution exploitation.

$$\zeta_\alpha = |\kappa_1.Z_\alpha - Z|, \zeta_\beta = |\kappa_2.Z_\beta - Z|, \zeta_\delta = |\kappa_3.Z_\delta - Z| \tag{2.14}$$

$$Z_1 = Z_\alpha - \mu_1.\zeta_\alpha, Z_2 = Z_\beta - \mu_2.\zeta_\beta, Z_3 = Z_\delta - \mu_3.\delta \tag{2.15}$$

$$Z(t+1) = \frac{Z_1 + Z_2 + Z_3}{3} \tag{2.16}$$

2.2.3. Attacking. The hunting process persists until the prey becomes immobile. In the mathematical model, the parameter a is gradually reduced at each iteration. Equation 2.17 specifies the control parameter a, where t indicates the current iteration and T represents the maximum iteration count.

$$a = 2 \times (1 - \frac{t}{T}) \tag{2.17}$$

2.3. Dynamic Weights Grey Wolf Optimization (Proposed algorithm)

In GWO, the position update of a search agent is given by three leading wolves, which are represented in Equation 2.16. This equal-weight strategy makes the algorithm static in nature and fails to differentiate between exploration and exploitation phases. The early iterations phase denotes exploration to avoid local optima, whereas the later iterations phase represents exploitation. To solve this issue, we use a dynamic weighting strategy that influences alpha by w_1, beta by w_2, and delta by w_3 as per the iteration "t".

$$w_1(t) + w_2(t) + w_3(t) = 1, \quad \forall\, t \in [0, T] \tag{2.18}$$

where T is the maximum number of iterations.

2.3.1. Cosine-Based Weight Evolution. A mathematically smooth transition from exploration to exploitation can be achieved using a cosine-based plan, which is represented in Equation 2.19.

$$f(t) = \frac{1 + \cos\left(\pi \frac{t}{T}\right)}{2} \tag{2.19}$$

The weights are then computed as:

Table 3: Parameters of algorithms and their respective values

Parameters	GWO	DW-GWO
No. of Wolf	25	25
Wolf length	No. of task	No. of task
No. of Iteration	500	500
a	[2,0]	[2,0]
μ	[-a,a]	[-a,a]
κ	[0,2]	[0,2]
w1	0.33	1 to 0.5
w2	0.33	0 to 0.34
w3	0.33	0 to 0.16
χ_1, χ_2	[0,1]	[0,1]

$$\begin{aligned} w_1(t) &= 0.5 + 0.5\, f(t), \\ w_2(t) &= 0.34\,\big(1 - f(t)\big), \\ w_3(t) &= 1 - w_1(t) - w_2(t) \end{aligned} \tag{2.20}$$

$$Z(t+1) = w_1(t) \times Z_1 + w_2(t) \times Z_2 + w_3(t) \times Z_3 \tag{2.21}$$

In the early iterations, the weights satisfy $w_1(t) \approx 1$, $w_2(t) \approx 0$, and $w_3(t) \approx 0$, which indicates strong alpha dominance and consequently enhanced exploration. At the latter of the iterations, the weights are $w_1(t) \approx 0.5$, $w_2(t) \approx 0.34$, and $w_3(t) \approx 0.16$. This stage is denoted as the exploitation. This dynamic transition ensures smooth and stable convergence, effectively reducing the risk of premature convergence while accelerating the convergence process during the exploitation phase. The Figure 1 represented a curve between weights and iterations. The Figure 2 and Algorithm 1 represent a flowchart and algorithm of the proposed algorithm, respectively.

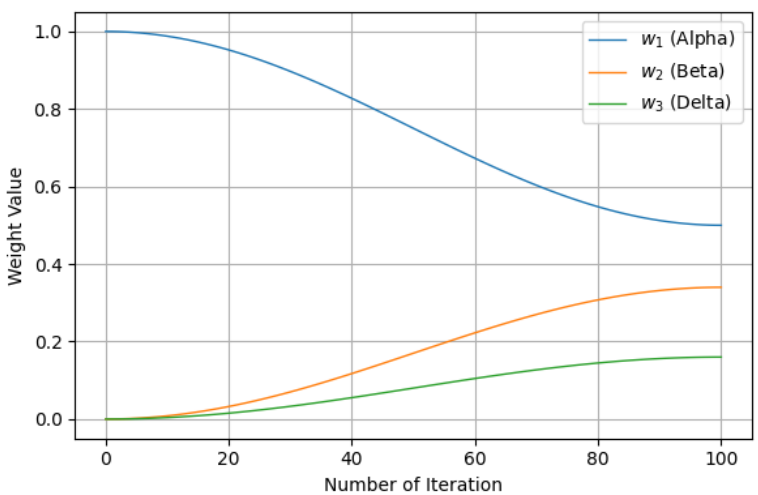

Figure 1: Curve for weights and iterations

3. Result Analysis

The proposed DW-GWO algorithm is tested across five different workflows to assess its performance in terms of TET and TEC based on the fitness function. The performance of the DW-GWO algorithm is evaluated against the ACO and GWO algorithms. The experimental evaluation considers five scientific

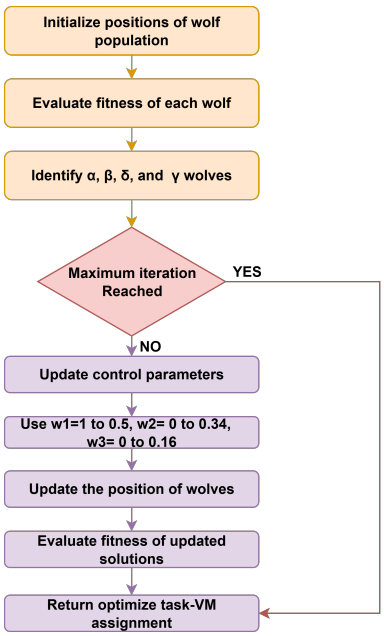

Figure 2: Flowchart of DW-GWO algorithm

Table 4: Simulation Parameters

Parameters	**Values**
No. of task	25 to 1000
No. of virtual machines	5
MIPS	1000
Bandwidth	1000
VM Policy	Time shared and space shared

workflows, such as CyberShake, Montage, Inspiral, SIPHT, and Epigenomics, with different task structures. The characteristics of the hardware system are as follows: an Intel(R) Core i5 processor running at 2.20 GHz, 4 GB of RAM, and a Windows 8 Pro 64-bit. All algorithms are implemented using the WorkflowSim 1.1 simulation toolkit with fixed workflow files and predefined virtual machines. Simulations used the default random settings of WorkflowSim, and run multiple times to ensure stable and consistent results. This simulator is an extended version of CloudSim. Table 3 presents the parameters utilized in the proposed method, while Table 4 denotes the simulator parameters.

3.1. Simulation Results

This section presents a comparative analysis of the proposed DW-GWO algorithm against existing approaches, namely ACO and GWO. The evaluation attention on the following performance indicators: TET and TEC. Experiments were conducted by varying the workflow size from 25 to 1000 tasks across five benchmark workflows: CyberShake, Inspiral, Montage, SIPHT, and Epigenomics. For all experiments, the maximum number of iterations was fixed at 500. Each algorithm along a particular scientific workflow is executed ten times, and the average values are recorded as results. The outcomes are provided in Tables 5, 6, and 7. The negative (-) values in these tables indicate an increase in TET and TEC compared to the proposed method.

Figure 3 (a) shows a comparison of the TEC of ACO, GWO, and the proposed DW-GWO algorithms under the CyberShake scientific workflow with the following task sizes: 30, 50, 100, and 1000. The proposed DW-GWO reduces TEC by 28.91%, 59.06%, 41.23%, and 39.12% compared to ACO, and by 4.14%, -0.38%, 1.78%, and 0.29% compared to GWO. These results demonstrate significant improvement

Table 5: Total execution cost with different workflows (In milliseconds)

					Percentage reduction	
Workflow	**Tasks**	**ACO**	**GWO**	**DW-GWO**	**From ACO**	**From GWO**
CyberShake	30	1623.46	1311.46	1222.494	28.91	4.14
	50	4846.28	3034.81	3046.62	59.06	-0.38
	100	11450.09	8251.61	8107.09	41.23	1.78
	1000	75465.22	54405.27	54243.26	39.12	0.29
Inspiral	30	8554.24	8159.96	7984.62	7.13	2.19
	50	14231.32	14063.63	13988.74	1.73	0.54
	100	28459.09	23830.97	23925.22	18.95	-0.39
	1000	252813.3	242255.5	241379.9	4.74	0.36
Montage	25	293.96	290.53	284.06	3.49	2.28
	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	11164.63	83.15	1.93
	60	31194.71	24784.39	22556.73	38.29	9.87
	100	45751.31	38485.94	36301.3	26.03	6.09
	1000	592117.6	494838.8	486010.7	21.83	1.82
Epigenomics	24	21593.16	18703.96	18528.98	16.53	0.94
	47	47942.12	44900.29	44890.38	6.80	0.03
	100	422283.1	414633.6	414971.5	1.76	-0.08
	997	4874893	3928125	3926024	24.17	0.05

Table 6: Total execution time with different workflows (In milliseconds)

					Percentage reduction	
Workflow	**Tasks**	**ACO**	**GWO**	**DW-GWO**	**From ACO**	**From GWO**
CyberShake	30	568.46	439.45	423.21	34.32	3.83
	50	1140.07	749.56	742.71	53.50	0.92
	100	2550.03	1798.48	1780.24	43.24	1.024
	1000	15565.76	11252.91	11198.41	38.99	0.48
Inspiral	30	1974.45	2515.54	2487.76	-20.63	1.11
	50	3627.57	3866.08	3770.74	-3.79	2.52
	100	7647.39	5665.73	5716.13	33.78	-0.88
	1000	61241.08	49755.23	49700.43	23.22	0.11
Montage	25	67.23	84.79	89.51	-24.89	-5.27
	50	136.45	153.49	144.89	-5.82	5.93
	100	278.44	290.79	282.43	-1.41	2.96
	1000	2640.57	2683.968	2682.39	-1.55	0.06
SIPHT	30	5400.3	4424.6	4419.29	22.19	0.12
	60	8338.89	6820.25	6302.37	32.31	8.21
	100	11009.13	9039.24	8945.41	23.07	1.05
	1000	120032.7	100337.1	98594.53	21.74	1.76
Epigenomics	24	6717.8	7846.05	7733.62	-13.13	1.45
	47	13140.41	13442.22	12800.82	2.65	5.01
	100	95277.34	121109.3	108561.3	-12.23	11.55
	997	1201141	876848.8	844381.4	42.25	3.84

Algorithm 1 DW-GWO workflow scheduling algorithm

Require: Workflow (V, D) and resources $\{VM_1, \ldots, VM_j\}$
Ensure: Assigning tasks to resources
Initialize parameters: $a \leftarrow 2$, $\mu \leftarrow 0$, $\kappa \leftarrow 0$, $\chi_1 \leftarrow \text{rand}(0,1)$, $\chi_2 \leftarrow \text{rand}(0,1)$
for $n \leftarrow 1$ **to** $popSize$ **do**
    $pop[n] \leftarrow$ randomize()
end for
Compute fitness for each individual using Equation (2.1).
$t \leftarrow 0$
while $t <$ maxIteration **do**
    Update α, β, δ (wolves) according to Equation (2.15).
    Update parameters μ, κ, a according to Equations (2.12), (2.13), (2.17).
    Update positions using Equation (2.21).
    Evaluate fitness and update best solutions (if improved).
    $t \leftarrow t + 1$
end while
return best mapping found

over ACO and slightly better or comparable performance to GWO. Figure 3 (b) illustrates the TET comparison of ACO, GWO, and DW-GWO for the CyberShake workflow. DW-GWO reduces TET by 34.32%, 53.50%, 43.24%, and 38.99% compared to ACO and by 3.83%, 0.92%, 1.02%, and 0.48% compared to GWO for considered tasks.

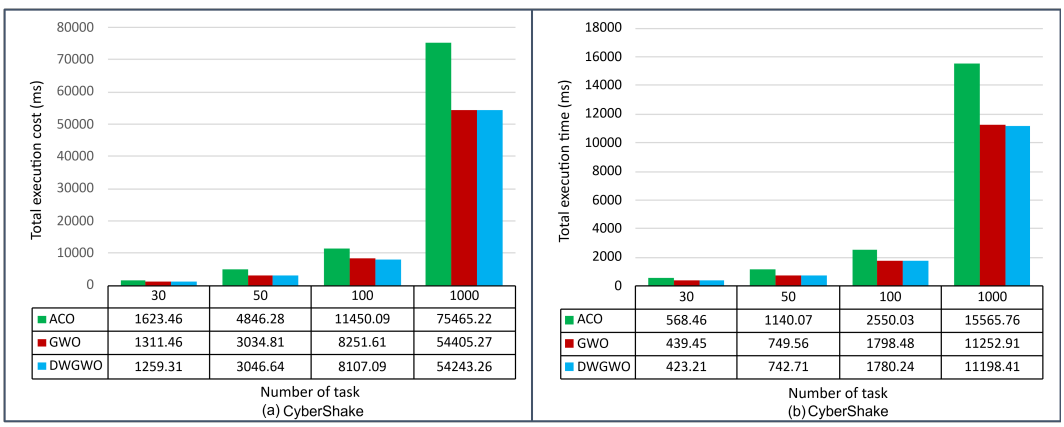

Figure 3: (a) TEC and (b) TET with CyberShake

Figure 4 (a) compares the execution cost of ACO, GWO, and DW-GWO approaches using the Inspiral workflow with the following task sizes: 30, 50, 100, and 1000. The proposed DW-GWO algorithm reduces TEC by 7.13%, 1.73%, 18.95%, and 4.73% compared to ACO, and by 2.19%, 0.53%, -0.39%, and 0.36% compared to GWO across the given task sizes. Figure 4 (b) compares the execution time of ACO, GWO, and DW-GWO approach using the Inspiral workflow. The DW-GWO algorithm reduces TET by -20.63%, -3.79%, 33.78%, and 23.22% compared to ACO and by 1.11%, 2.52%, -0.88%, and 0.11% compared to GWO for considered tasks.

Figure 5 (a) presents the Montage workflow results. DW-GWO reduces TEC by 3.48%, -1.42%, 1.33%, and 1.13% compared to ACO, and 2.27%, -1.51%, 1.85%, and 0.99% compared to GWO for 25, 50, 100, and 1000 tasks, respectively. Improvement is moderate, particularly for large-scale workloads. Figure 5 (b) compares the execution time of ACO, GWO, and DW-GWO methods using the Montage workflow. The DW-GWO algorithm shows -24.89%, -5.82%, -1.41%, and -1.55% reduction compared to ACO and -5.27%, 5.93%, 2.96%, and 0.05% compared to GWO for 25, 50, 100, and 1000 tasks.

Table 7: Fitness with different workflows (In milliseconds)

Workflow	Tasks	ACO	GWO	DW-GWO
CyberShake	30	1095.96	875.455	841.26
	50	2993.17	1892.185	1894.67
	100	7000.06	5025.04	4943.665
	1000	45515.49	32829.09	32720.84
Inspiral	30	5264.34	5337.75	5236.19
	50	8929.44	8964.855	8879.74
	100	18053.24	14748.35	14820.68
	1000	157027.2	146005.4	145540.2
Montage	25	180.59	187.66	186.785
	50	383.79	392.025	392.57
	100	791.35	800.885	784.79
	1000	7706.78	7719.80	7656.32
SIPHT	30	12924.31	7902.445	7791.962
	60	19766.8	15802.32	14429.55
	100	28380.22	23762.59	22623.36
	1000	356075.2	297587.9	292302.6
Epingenomics	24	14155.48	13275.01	13131.3
	47	30541.27	29171.26	28845.6
	100	258780.2	267871.4	261766.4
	997	3038017.48	2402487.14	2385203.01

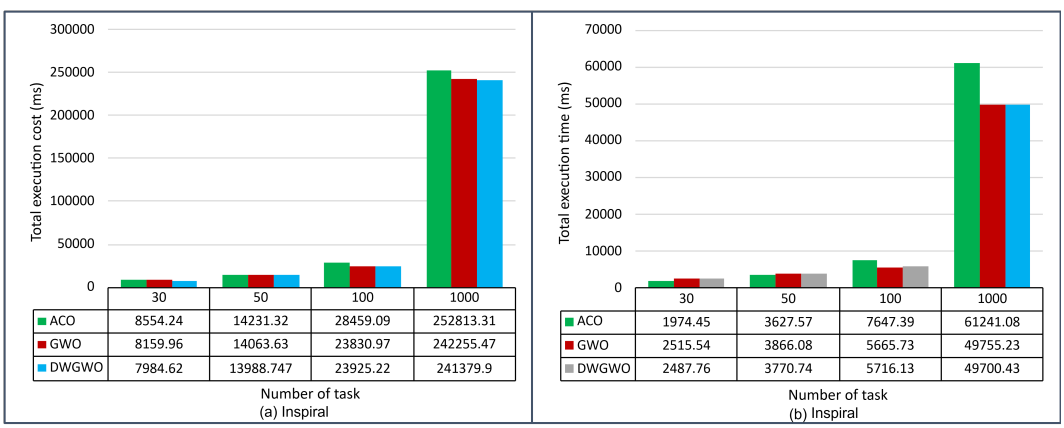

Figure 4: (a) TEC and (b) TET with Inspiral

Figure 6 (a) compares the TEC of ACO, GWO, and DW-GWO using the SIPHT workflow with following task sizes: 30, 60, 100, and 1000. The DW-GWO achieves reductions of 83.15%, 38.29%, 26.03%, and 21.83% over ACO, and 1.93%, 9.87%, 6.01%, and 1.81% over GWO for task sizes 30, 60, 100, and 1000, respectively. The reduction is most prominent for small and medium scientific workflows. Figure 6 (b) illustrates TET for SIPHT workflow. DW-GWO achieves reductions of 22.19%, 32.31%, 23.07%, and 21.74% over ACO and 0.12%, 8.21%, 1.04%, and 1.76% over GWO for 30, 60, 100, and 1000 tasks, respectively.

Figure 7 (a) illustrates TEC results for the Epigenomics workflow. DW-GWO achieves 16.53%, 6.79%, 1.76%, and 24.16% cost reduction over ACO, and 0.94%, 0.02%, -0.08%, and 0.05% over GWO for 24, 47, 100, and 997 tasks. DW-GWO consistently performs better than ACO and remains competitive with GWO on large datasets. Figure 7 (b) compares the TEC of ACO, GWO, and DW-GWO methods under the Epigenomics workflow. In the Epigenomics workflow, DW-GWO achieves -13.13%, 2.65%, -12.23%,

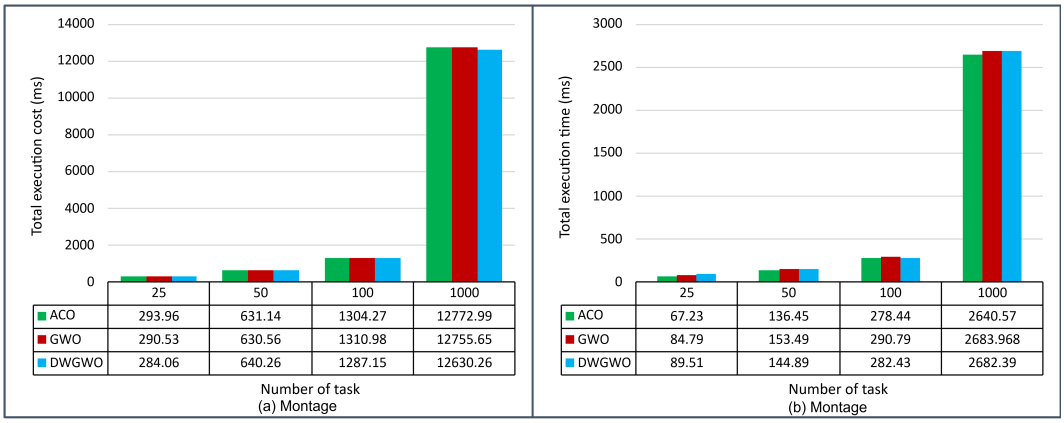

Figure 5: (a) TEC and (b) TET with Montage

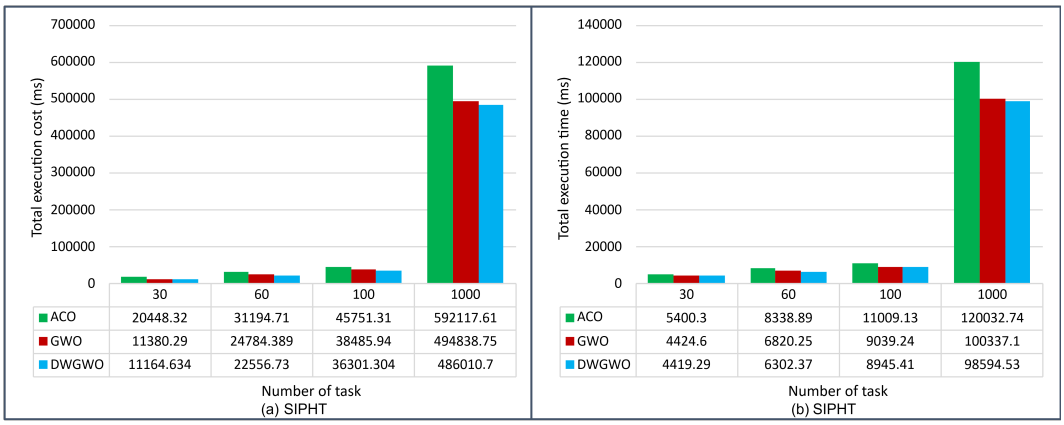

Figure 6: (a) TEC and (b) TET with SIPHT

and 42.25% improvement compared to ACO and 1.45%, 5.01%, 11.55%, and 3.84% compared to GWO.

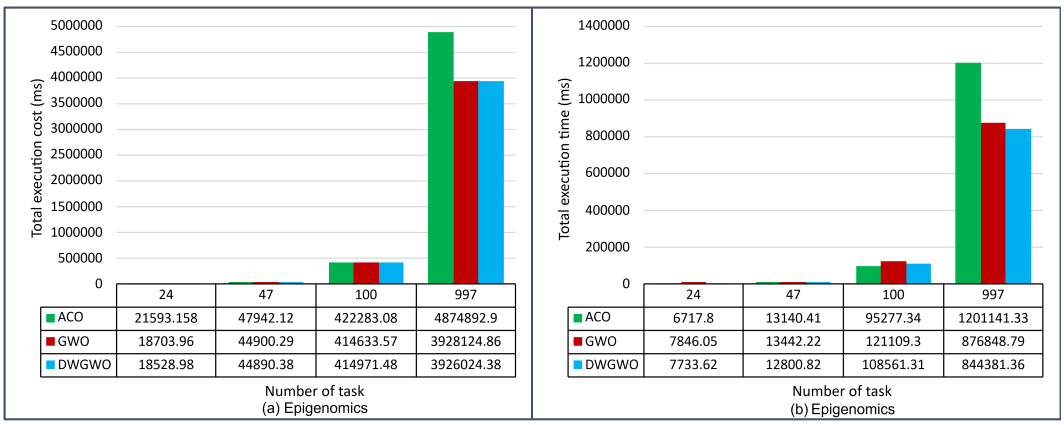

Figure 7: (a) TEC and (b) TET with Epigenomics

4. Conclusion and future scope

The proposed DW-GWO algorithm shows significant improvements in workflow scheduling compared to the basic GWO and ACO algorithms. It reduces both TEC and TET across multiple scientific workflows such as CyberShake, SIPHT, Inspiral, Montage, and Epigenomics, tested at different task sizes. The performance improvement is achieved by using dynamic weights for the α, β, and δ wolves, rather than equal weighting, in basic GWO, where all leaders are influenced by equal weight. The DW-GWO assigns a weight of 1.0 to 0.5 for α, 0 to .34 for β, and 0 to 0.16 for δ. This approach enhances the exploration ability in the early phase and exploitation in the later phase, resulting in faster convergence, avoidance of local optima, and improved solution quality.

In terms of TEC, DW-GWO achieved substantial reductions over ACO, reaching up to 83.15% for the SIPHT workflow with 30 tasks, 59.06% for CyberShake with 50 tasks, and 24.17% for Epigenomics with 997 tasks, while also outperforming GWO by up to 9.87% for SIPHT with 60 tasks. Similarly, for TET, DW-GWO reduced execution time by up to 53.50% over ACO for CyberShake with 50 tasks and 42.25% for Epigenomics with 997 tasks, along with consistent improvements over GWO of up to 8.21% across medium and large workflows. These results confirm the effectiveness of DW-GWO in reducing both execution cost and execution time compared to the baseline algorithms. The proposed DW-GWO algorithm is evaluated only on static scientific workflows, and its performance for dynamic or online workflow scheduling is not considered in this study. In the future, the method can be improved by making weights adaptive, combining it with machine learning or other algorithms to make more intelligent decisions, and also reducing energy consumption, along with costs and time.

Acknowledgements

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. M. N. O. Sadiku, S. M. Musa and O. D. Momoh, *Cloud computing: opportunities and challenges*, IEEE Potentials **33:1** (2014), 34–36.

2. S. Bansal, M. Aggarwal and H. Aggarwal, *Advancements and applications in fog computing*, Security Designs for the Cloud, IoT, and Social Networking, pp. 207–240, Wiley Online Library (2019).

3. S. Mangalampalli, K. S. Pokkuluri, R. Kocherla, A. Rapaka and N. R. Kota, *An efficient workflow scheduling algorithm in cloud computing using cuckoo search and PSO algorithms*, Innovations in Computer Science and Engineering: Proceedings of the Ninth ICICSE, 2021, pp. 137–145, Springer (2022).

4. N. A. Alawad and B. H. Abed-alguni, *Discrete island-based cuckoo search with highly disruptive polynomial mutation and opposition-based learning strategy for scheduling of workflow applications in cloud environments*, Arabian Journal for Science and Engineering, vol. 46, no. 4, pp. 3213–3233, Springer (2021).

5. W.-N. Chen and J. Zhang, *An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 39, no. 1, pp. 29–43, IEEE (2008).

6. S. Mangalampalli, K. S. Pokkuluri, G. N. Satish, and K. V. R. Kumar, *An effective workflow scheduling algorithm in cloud computing using cat swarm optimization*, ECS Transactions, vol. 107, no. 1, p. 2523, IOP Publishing (2022).

7. A. M. Manasrah and H. Ba Ali, *Workflow scheduling using hybrid GA-PSO algorithm in cloud computing*, Wireless Communications and Mobile Computing, vol. 2018, no. 1, p. 1934784, Wiley Online Library (2018).

8. Y. Ge and G. Wei, *GA-based task scheduler for the cloud computing systems*, in *Proceedings of the 2010 International Conference on Web Information Systems and Mining*, vol. 2, pp. 181–186, IEEE (2010).

9. M. A. Tawfeek, A. El-Sisi, A. E. Keshk, and F. A. Torkey, *Cloud task scheduling based on ant colony optimization*, in *Proceedings of the 2013 8th International Conference on Computer Engineering & Systems (ICCES)*, pp. 64–69, IEEE (2013).

10. T. Deepa and D. Cheelu, *A comparative study of static and dynamic load balancing algorithms in cloud computing*, in *Proceedings of the 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pp. 3375–3378, IEEE (2017).

11. N. Bansal and A. K. Singh, *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).

12. N. Bacanin, M. Zivkovic, T. Bezdan, K. Venkatachalam, and M. Abouhawwash, *Modified firefly algorithm for workflow scheduling in cloud-edge environment*, Neural Computing and Applications, vol. 34, no. 11, pp. 9043–9068, Springer (2022).

13. M. Dorigo, M. Birattari, and T. Stützle, *Ant colony optimization*, *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39 (2007), IEEE.

14. T. Bezdan, M. Zivkovic, M. Antonijevic, T. Zivkovic, and N. Bacanin, *Enhanced flower pollination algorithm for task scheduling in cloud computing environment*, in *Machine Learning for Predictive Analysis: Proceedings of ICTIS 2020*, pp. 163–171, Springer (2020).

15. S. Raghavan, P. Sarwesh, C. Marimuthu, and K. Chandrasekaran, *Bat algorithm for scheduling workflow applications in cloud*, in *Proceedings of the 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV)*, pp. 139–144, IEEE (2015).

16. A. Y. Hamed, M. Kh. Elnahary, and H. H. El-Sayed, *Optimization task scheduling bee colony algorithm for heterogeneous cloud computing systems*, *Appl. Math*, vol. 16, no. 6, pp. 899–909 (2022).

17. B. Sahu, S. K. Swain, S. Mangalampalli, and S. Mishra, *Multiobjective Prioritized Workflow Scheduling in Cloud Computing Using Cuckoo Search Algorithm*, *Applied Bionics and Biomechanics*, vol. 2023, no. 1, p. 4350615, Wiley Online Library (2023).

18. P. Zhang, S. Shu, and M. Zhou, *An online fault detection model and strategies based on SVM-grid in clouds*, *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 445–456, IEEE (2018).

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

20. H. Li, D. Wang, J. R. Canizares Abreu, Q. Zhao, and O. Bonilla Pineda, *PSO+ LOA: hybrid constrained optimization for scheduling scientific workflows in the cloud*, *The Journal of Supercomputing*, vol. 77, pp. 13139–13165, Springer (2021).

21. N. Arora and R. K. Banyal, *Workflow scheduling using particle swarm optimization and gray wolf optimization algorithm in cloud computing*, *Concurrency and Computation: Practice and Experience*, vol. 33, no. 16, p. e6281, Wiley Online Library (2021).

22. S. K. Bothra, S. Singhal, and H. Goyal, *Cost effective hybrid genetic algorithm for workflow scheduling in cloud*, *System Research and Information Technologies*, no. 3, pp. 121–138 (2022).

23. S. S. Hammed and B. Arunkumar, *A cost effective-secure algorithm for work-flow scheduling in cloud computing*, *Internet Technology Letters*, vol. 6, no. 1, p. e233, Wiley Online Library (2023).

24. M. Mokni, S. Yassa, J. E. Hajlaoui, M. N. Omri, and R. Chelouah, *Multi-objective fuzzy approach to scheduling and offloading workflow tasks in Fog–Cloud computing*, *Simulation Modelling Practice and Theory*, vol. 123, p. 102687, Elsevier (2023).

25. Jinchao Chen, Pengcheng Han, Ying Zhang, Tao You, and Pengyi Zheng, *Scheduling energy consumption-constrained workflows in heterogeneous multi-processor embedded systems*, *Journal of Systems Architecture*, vol. 142, p. 102938, Elsevier (2023).

26. Jinchao Chen, Pengcheng Han, Yifan Liu, and Xiaoyan Du, *Scheduling independent tasks in cloud environment based on modified differential evolution*, *Concurrency and Computation: Practice and Experience*, vol. 35, no. 13, p. e6256, Wiley Online Library (2023).

27. Hind Mikram, Said El Kafhali, and Youssef Saadi, *HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment*, *Simulation Modelling Practice and Theory*, vol. 130, p. 102864, Elsevier (2024).

28. 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).

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

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

31. 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).

32. 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.

33. D. Ouelhadj, J. Garibaldi, J. MacLaren, R. Sakellariou, and K. Krishnakumar, *A multi-agent infrastructure and a service level agreement negotiation protocol for robust scheduling in grid computing*, *Proc. European Grid Conference*, pp. 651–660 (2005), Springer.

34. U. Schwiegelshohn and R. Yahyapour, *Analysis of first-come-first-serve parallel job scheduling*, *Proc. SODA*, vol. 98, pp. 629–638 (1998).

35. R. V. Rasmussen and M. A. Trick, *Round robin scheduling–a survey*, *European Journal of Operational Research*, vol. 188, no. 3, pp. 617–636 (2008), Elsevier.

36. R. Skinderowicz, *Implementing population-based ACO*, *Proc. International Conference on Computational Collective Intelligence*, pp. 603–612 (2014), Springer.

37. G. Patel, R. Mehta, and U. Bhoi, *Enhanced load balanced min-min algorithm for static meta task scheduling in cloud computing*, *Procedia Computer Science*, vol. 57, pp. 545–553 (2015), Elsevier.

38. 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

and

Vaishali Maheshwari,
Shri vishwakarma skill university , Haryana
India.
E-mail address: vaishalimaheshwari@svsu.ac.in