Available online at http://jprm.sms.edu.pk/
Journal of Prime Research in Mathematics, 2026, 256–268

Journal of Prime Research in Mathematics

Workflow Scheduling in Cloud Computing Using Customised Weights Grey Wolf Optimization

Chotu Lal[a,*], Harish Sharma[a]

[a] *Department of Computer Science and Engineering, Rajasthan Technical University,Rawat Bhata Road, Kota, Rajasthan, India,*

Abstract

This study proposes a workflow scheduling approach that incorporates varying service quality demands in cloud environments. The main target of scheduling in the cloud is to minimise total execution time (TET) and total execution cost (TEC) while ensuring compliance with the service level agreement (SLA). Workflow scheduling is an NP-hard problem (Nondeterministic Polynomial time) that presents significant challenges in cloud computing systems. Traditional algorithms cannot efficiently solve the workflow scheduling problem in polynomial time. This paper presents a multi-objective system for workflow scheduling using a customised weights grey wolf optimization (CWGWO) approach in cloud computing. The outcome of this algorithm is effectively balances between exploration and exploitation capabilities. It is designed to minimise the TET and TEC when tasks are interdependent. Experimental results indicate that the CWGWO algorithm outperforms both the Ant Colony Optimisation (ACO) and Grey Wolf Optimisation (GWO) algorithms in terms of TET and TEC.

Keywords: Optimization Algorithms, Metaheuristic Algorithms, Customised Weights Grey Wolf Optimization, Cloud Computing, Workflow Scheduling, Nature-Inspired Algorithms

1. Introduction

The cloud system provides the following services: scalable storage, computing power, and networking resources are available online. These services enable the system to be flexible and efficient. It allows many users to choose specific services with flexibility on demand, similar to a subscription model. The field of cloud computing has expanded, providing numerous benefits to users [1]. These benefits are offered by a service provider through virtualisation, where tasks are allocated and executed on virtual machines (VMs) in a process called task scheduling. The task scheduling in cloud environments is NP-hard, where NP stands for Nondeterministic Polynomial time, due to their dynamic and heterogeneous nature [2]. Workflow scheduling is a process that aligns tasks with virtual machines when those tasks are interdependent.

Several studies have proposed algorithms to minimize workflow makespan and cost [3, 5, 6, 4, 7]; however, there is still scope for achieving further reductions in both metrics. The work in [4] introduces a method

*Chotu Lal
Email addresses: clal.phd22@rtu.ac.in (Chotu Lal), hsharma@rtu.ac.in (Harish Sharma)

Received : 31 August 2025*; Accepted:* 03 June 2026*; Published Online:* 11 July 2026

capable of processing both balanced and unbalanced workflows, while simultaneously optimizing communication and data-transfer related costs. The DiCSPM approach enhances data-transfer performance and reduces communication costs when compared with GWO, BRS, and PSO [8]. The genetic algorithm (GA) requires a significant amount of time to achieve optimal solutions . Tawfeek et al. [9] applied an ant colony optimisation (ACO) based scheduling method for task scheduling and reported that it achieved a shorter makespan than the first-come, first-served (FCFS) and round-robin (RR) scheduling strategies. However, the ACO algorithm is complex and takes a substantial amount of time to reach an optimal solution [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 algorithm is used for cloud task scheduling to reduce makespan and shows improved performance compared to PBACO, ACO, Min-Min, and FCFS strategies [13]. The paper uses the Bat Algorithm, a metaheuristic inspired by bat echolocation, to reduce execution cost [14]. This paper uses the artificial bee colony (ABC) method to schedule tasks to improve load balancing and resource utilisation [15]. The Cuckoo Search algorithm is used in workflow scheduling to improve energy efficiency and reduce SLA violations in cloud computing [16]. The cat swarm optimisation (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 Customised Weights Grey Wolf Optimization (CWGWO) algorithm. In the original GWO, the top three leaders are influenced by equal importance. In the proposed approach (CWGWO), a customised weight is adopted for alpha, which is influenced by weight w_1=0.50, beta is influenced by w_2 = 0.34, and delta is influenced by w_3 =0.16. These weights are assigned to these leaders. In CWGWO, 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 diversity. The CWGWO aims to improve exploration-exploitation balance, accelerate convergence, and avoid premature trapping in local optima. The proposed (CWGWO) algorithm is used in workflow scheduling to reduce TET and TEC. The workflow scheduling algorithm was simulated using WorkflowSim.

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 CWGWO 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.

Table 1: Study of task scheduling algorithms in cloud Computing

References	Approaches	Used Workflows	Key Contribution	Limitations
2018 [17]	BBO and HMBBO	4	It converges more rapidly than BBO, PSO, and HEFT.	The approach does not consider both the distribution of tasks over resources and the associated expenses.
2021 [18]	Multi-objective optimization algorithm	2	The proposed method achieves a maximum 50% reduction in energy consumption relative to MOPT, EM-MOO, and PEFT.	The results compare only cybershake and montage workflows.
2021 [19]	PSO + LOA	3	The performance is better in TEC compared to PSO and LOA.	Only discussed a single parameter that is cost.
2021 [20]	PSO-GWO	4	Effectively reducing execution cost and time compare to GWO and PSO.	The results do not consider the scalability of resources with large-scale workflows.
2022 [21]	CHGA algorithm	4	CHGA is reduced cost compare to considered algorithms.	Only focused cost parameter.
2023 [22]	Cost-effective algorithm	3	Reduced execution cost compare to PSO and GA.	Only three scientific workflows used.
2023 [24]	Population based approach	0	Reduced execution time and cost by allocated task to their appropriate resource.	No scientific workflows used.
2023 [23]	Novel energy coefficient based algorithm	1	Improvement in energy consumption and data dependency management.	The focus was solely on energy consumption, and the results were compared with only two workflows.
2024 [25]	HEPGA algorithm	0	Better at optimizing tasks and fitness values.	Focused solely on minimizing makespan.
2025 [26]	MGWO	2	This study employs MGWO for multi-objective workflow scheduling.	Applied on only two workflows used.
2025 [27]	HPWOA	3	Hybrid HPWOA for single-objective workflow scheduling.	Only three scientific workflows used and discussed only the cost parameter.
2025 [28]	PSO-MGWO	4	This study is used for Hybrid multi-objective workflow scheduling.	Only four scientific workflows used.
	CWGWO (proposed approach)	5	Reduce execution time and execution cost	This approach is not used for energy consumption and other objectives.

2.1.1. ***Total Execution Time***

As described in [19], the computation time (CMT) of task t_q on resource r_j is given by Eq. (2.2).

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

Here, $Length_q$ denotes the instruction size of task T_q in million instructions (MI), and PC_j refers to the

processing capability of resource r_j, measured in MFLOPS. 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 from all the input files. The Eq. (2.3) denoted the transferring of the task (T_q).

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

Here, the task T_p is a predecessor task of task T_q, $Data_{(T_p,T_q)}$ is the data size to be transferred from task T_p to T_q, and BW for bandwidth. If T_p and T_q share the same VM, their transfer time is zero. Eq. (2.4) defines the processing time (PCT) of task T_q on VM r_j.

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

The start time (STT) of task T_q on VM r_j is determined by Eq. (2.5).

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

The finish time (FinT) of task T_q on VM r_j is computed by Eq. (2.6).

$$FinT(T_q, r_j) = \begin{cases} PCT(T_q, r_j), & \text{if prede } (T_q) = 0 \\ \max_{T_p \in t_q} \{ FinT(T_p) + PCT(T_p, r_j) \} & \text{if prede } (T_q) \neq 0 \end{cases} \tag{2.6}$$

The total execution time of a directed acyclic graph (DAG) is taken as the largest finish time among its tasks.

$$TET = \max_{T_q \in T} \{ FinT(T_q, r_j) \} \tag{2.7}$$

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 apply a fixed charge for moving each unit of data between two VM nodes [19]. The execution cost can be defined by the Eq. (2.8).

$$EXT(r_j) = \lfloor \frac{RFT_j - RST_j}{\tau} \rfloor * vc_j \tag{2.8}$$

Here, vc_j is the cost of leasing VM r_j per time unit, and τ represents the associated time interval. RFT_j and RST_j represent r_j's finish renting time and start renting time while executing a workflow, respectively. $\lfloor \rfloor$ denotes the floor function, which rounds a value down to the nearest integer. The TEC of the workflow is denoted by Eq. (2.9).

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

Where n is the number of virtual machines. This work aims to assign each workflow task to a suitable VM while minimizing makespan and keeping the TEC within budget.

$$Min \;\; i.e. \, TEC \leq Budget \tag{2.10}$$

2.2. Grey wolf optimisation algorithm

Mirjalili et al. [29] presented GWO, is a metaheuristic modeled on the behavior of grey wolf packs. In GWO, the wolves stay in a hierarchical order, such as alpha, beta, delta, and omega. The following stages illustrate the algorithm.

2.2.1. Encircling prey

In GWO, grey wolves surround their prey while hunting.

$$\zeta = |\kappa.Z_P(t) - Z(t)| \tag{2.11}$$

$$Z(t+1) = Z_P(t) - \mu.\zeta \tag{2.12}$$

Eq. (2.11) and Eq. (2.12) help in updating the position of the wolves according to iteration (t). Z_p and Z denote the position of prey and the current location of wolf, respectively. The values of μ and κ are obtained using Eq. (2.13) and Eq. (2.14), respectively.

$$\mu = 2.a.\chi_1 - a \tag{2.13}$$

$$\kappa = 2.\chi_2 \tag{2.14}$$

The random variables χ_1 and χ_2 lie within the interval $[0, 1]$, while Eq. (2.18) defines the linear decrement of a from 2 to 0.

2.2.2. Hunting

In the hunting process, the alpha plays the most critical role by guiding the beta and delta, and encircles the prey. The dominant wolves locate the prey, representing the optimal solution. Remaining agents update their locations with respect to the dominant wolves to exploit the best solutions.

$$\zeta_\alpha = |\kappa_1.Z_\alpha - Z|, \zeta_\beta = |\kappa_2.Z_\beta - Z|, \zeta_\delta = |\kappa_3.Z_\delta - Z| \tag{2.15}$$

$$Z_1 = Z_\alpha - \mu_1.\zeta_\alpha, Z_2 = Z_\beta - \mu_2.\zeta_\beta, Z_3 = Z_\delta - \mu_{3\cdot\delta} \tag{2.16}$$

$$Z(t+1) = \frac{Z_1 + Z_2 + Z_3}{3} \tag{2.17}$$

2.2.3. Attacking

The grey wolf continues the hunt until it ceases to move. During each iteration, a gradually decreases. In Eq. (2.18), a is the controlling variable, ι denotes the current iteration, and τ is the total number of iterations.

$$a = 2 \times (1 - \frac{\iota}{\tau}) \tag{2.18}$$

2.3. Customized weight grey wolf optimization (CWGWO)

In the normal GWO, the main three wolves: α, β and δ, guide the other wolves with equal weights, where $w_1 = w_2 = w_3 = \frac{1}{3}$. But in reality, the top leader α has more control than β, and last δ. In the proposed approach, we use the weights assigned to wolves leader as constant values likely, $w_1 = \frac{3}{6}$ for α, $w_2 = \frac{2}{6}$ for β, and $w_3 = \frac{1}{6}$ for δ. This way, α becomes the main guide in the start and middle of the search, β gives medium support, and δ plays a smaller helping role. So this weighting enhances convergence capability by maintaining strong exploration in the early phase while improving exploitation in the latter phase. The Figure 1 shows the Flowchart of the CWGWO algorithm. The Algorithm 1 represents the pseudocode of the CWGWO workflow scheduling algorithm.

$$Z(t+1) = \frac{3}{6} \times Z_1 + \frac{2}{6} \times Z_2 + \frac{1}{6} \times Z_3 \tag{2.19}$$

Table 2: Parameters of algorithms and their respective values

Parameters	**GWO**	**CWGWO**
Wolf Size	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	0.5
w2	0.33	0.34
w3	0.33	0.16
χ_1, χ_2	[0,1]	[0,1]

Table 3: Simulation Parameters

Parameters	**Values**
Task	24 to 1000
Virtual machines	5
MIPS	1000
RAM	512
Bandwidth	1000
Processor	1
VM Policy	Time shared and space shared

Algorithm 1 The CWGWO Workflow Scheduling

Input: Workflow and set of resources
Output: Task allocation on virtual machines
Initialize: $a \leftarrow 2$, $\mu \leftarrow 0$, $\kappa \leftarrow 0$, $\chi_1 \leftarrow rand[0,1]$, $\chi_2 \leftarrow rand[0,1]$
for $s = 0$ **to** $populationSize$ **do**
    $pop[s] \leftarrow randomize()$
end for
Compute the fitness function using Eq. (2.1).
$s \leftarrow 0$
while $s < maximumIteration$ **do**
    Update α, β, and δ wolves as per Eq. (2.16).
    Update parameters μ, κ, and a as per Eq. (2.13), (2.14), and (2.18).
    Update positions as per Eq. (2.19).
    $s \leftarrow s + 1$
end while
Return: Optimal mapping of tasks to virtual machines

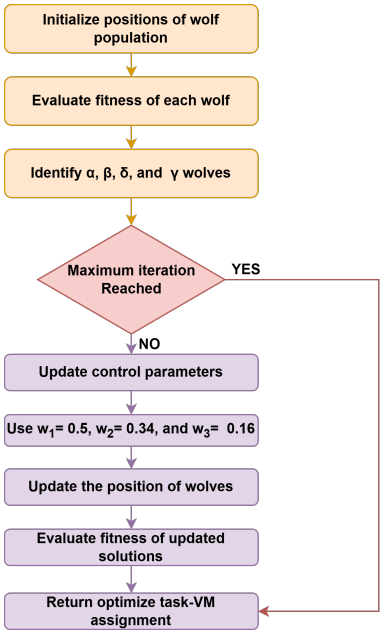


Figure 1: Flowchart of CWGWO algorithm

3. Results and discussion

The proposed CWGWO 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 CWGWO algorithm is evaluated against the ACO and GWO algorithms. The experimental evaluation considers five scientific workflows, such as CyberShake, Montage, Inspiral, SIPHT, and Epigenomics, with different task structures. The characteristics of the hardware system are as follows: Core i5 processor, 4 GB RAM, and Windows 8. These algorithms are implemented on a standard tool known as WorkflowSim 1.1 simulation toolkit [30]. This simulator is an extended version of CloudSim. Table 2 lists the algorithm parameters, while Table 3 denotes the simulator parameters.

This section presents a comparative discussion of the CWGWO algorithm against existing approaches, namely ACO and GWO. The evaluation focuses on two performance metrics: 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 4, 5, and 6.

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

					Percentage reduction	
Workflow	**Task**	**ACO**	**GWO**	**CWGWO**	**From ACO**	**From GWO**
CyberShake	30	568.46	439.45	433.21	29.36	1.44
	50	1140.07	749.56	745.31	52.09	0.57
	100	2550.03	1798.48	1785.48	41.78	0.73
	1000	15565.76	11252.91	11201.01	38.33	0.46
SIPHT	30	5400.3	4424.6	4422.29	22.05	0.06
	60	8338.89	6820.25	6309.37	22.2	8.09
	100	11009.13	9039.24	9001.41	21.79	0.42
	1000	120032.74	100337.1	98609.21	19.62	1.75
Inspiral	30	1974.45	2515.54	2510.16	-21.50	0.21
	50	3627.57	3866.08	3809.4	-6.16	1.48
	100	7647.39	5665.73	5720.13	34.97	-0.95
	1000	61241.08	49755.23	49712.38	23.08	0.08
Montage	25	67.23	84.79	87.02	-20.70	-2.57
	50	136.45	153.49	149.82	-11.10	2.45
	100	278.44	290.79	285.49	-4.24	1.85
	1000	2640.57	2683.968	2686.39	-1.61	-0.09
Epigenomics	24	6717.8	7846.05	7709.11	-14.37	1.77
	47	13140.41	13442.22	13040.21	-2.24	3.08
	100	95277.34	121109.3	108661.17	-21.38	11.45
	997	1201141.33	876848.79	854382.26	36.98	2.63

Table 4: Total execution cost with different workflows (In milliseconds)

					Percentage reduction	
Workflow	**Task**	**ACO**	**GWO**	**CWGWO**	**From ACO**	**From GWO**
CyberShake	30	1623.46	1311.46	1287.42	23.79	1.86
	50	4846.28	3034.81	3032.74	59.68	0.07
	100	11450.09	8251.61	8112.17	38.76	1.718
	1000	75465.22	54405.27	54256.26	38.70	0.27
SIPHT	30	20448.32	11380.29	11271.69	79.68	0.96
	60	31194.71	24784.389	23711.02	25.86	4.52
	100	45751.31	38485.94	37327.56	18.87	3.10
	1000	592117.6	494838.75	488020.1	19.65	1.39
Inspiral	30	8554.24	8159.96	7899.61	4.83	3.29
	50	14231.32	14063.63	13997.71	1.19	0.47
	100	28459.09	23830.97	23901.22	19.42	-0.29
	1000	252813.3	242255.47	241560.8	4.35	0.28
Montage	25	293.96	290.53	286.38	1.18	1.44
	50	631.14	630.56	628.26	0.09	0.36
	100	1304.27	1310.98	1308.31	-0.51	0.20
	1000	12772.99	12755.65	12722.17	0.14	0.26
Epigenomics	24	21593.16	18703.96	18601.14	15.44	0.55
	47	47942.12	44900.29	44860.69	6.78	0.9
	100	422283.1	414633.57	414601.6	1.85	0.08
	997	4874893	3928124.86	3927127	24.11	0.03

Table 6: Fitness with different workflows (In milliseconds)

Workflow	Task	ACO	GWO	CWGWO
CyberShake	30	1095.96	875.455	860.315
	50	2993.175	1892.185	1889.025
	100	7000.06	5025.045	4948.825
	1000	45515.49	32829.09	32728.64
SIPHT	30	12924.31	7902.445	7846.99
	60	19766.8	15802.32	15010.2
	100	28380.22	23762.59	23164.49
	1000	356075.2	297587.9	293314.7
Inspiral	30	5264.345	5337.75	5204.885
	50	8929.445	8964.855	8903.555
	100	18053.24	14748.35	14810.68
	1000	157027.2	146005.4	145636.6
Montage	25	180.595	187.66	186.7
	50	383.795	392.025	389.04
	100	791.355	800.885	796.9
	1000	7706.78	7719.809	7704.28
Epigenomics	24	14155.48	13275.01	13155.13
	47	30541.27	29171.26	28950.45
	100	258780.2	267871.4	261631.4
	997	3038017	2402487	2390755

Figure 2 (a) shows the TEC under the CyberShake workflow with the following task sizes: 30, 50, 100, and 1000. The proposed CWGWO reduces TEC by 23.79%, 59.68%, 38.76%, and 38.70% compared to ACO, and by 1.86%, 0.07%, 1.71%, and 0.27% compared to GWO. These results demonstrate significant improvement over ACO and slightly better or comparable performance to GWO. Figure 2 (b) shows the TET for CyberShake workflow. CWGWO reduces TET by 29.36%, 52.09%, 41.78%, and 38.33% compared to ACO and by 1.44%, 0.57%, 0.73%, and 0.46% compared to GWO for considered tasks, respectively.

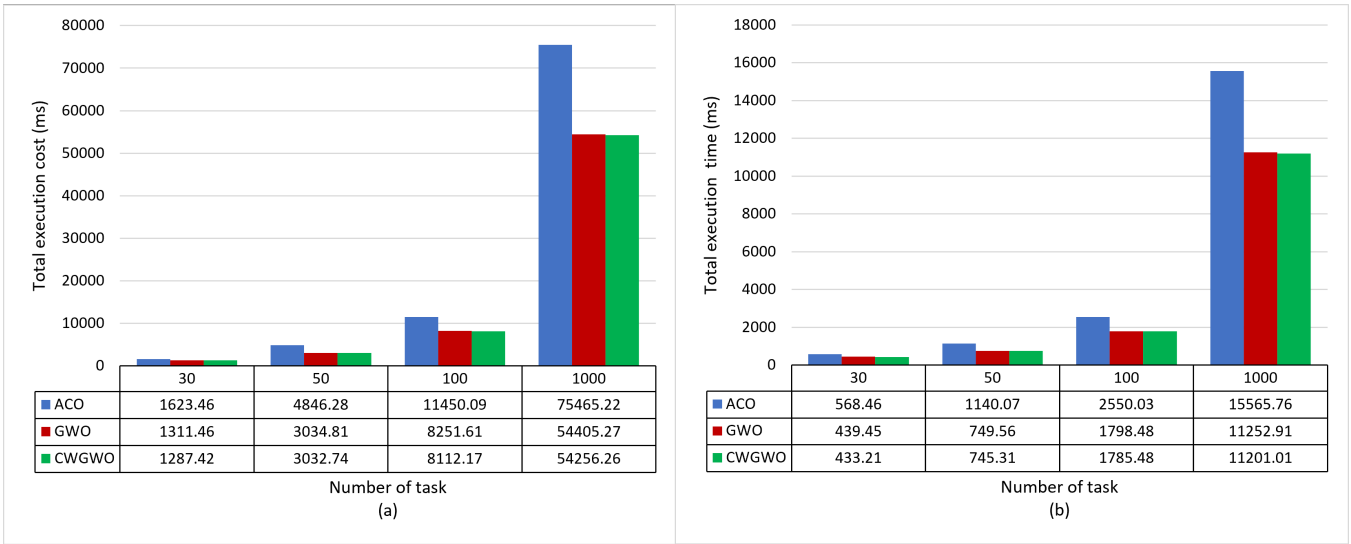
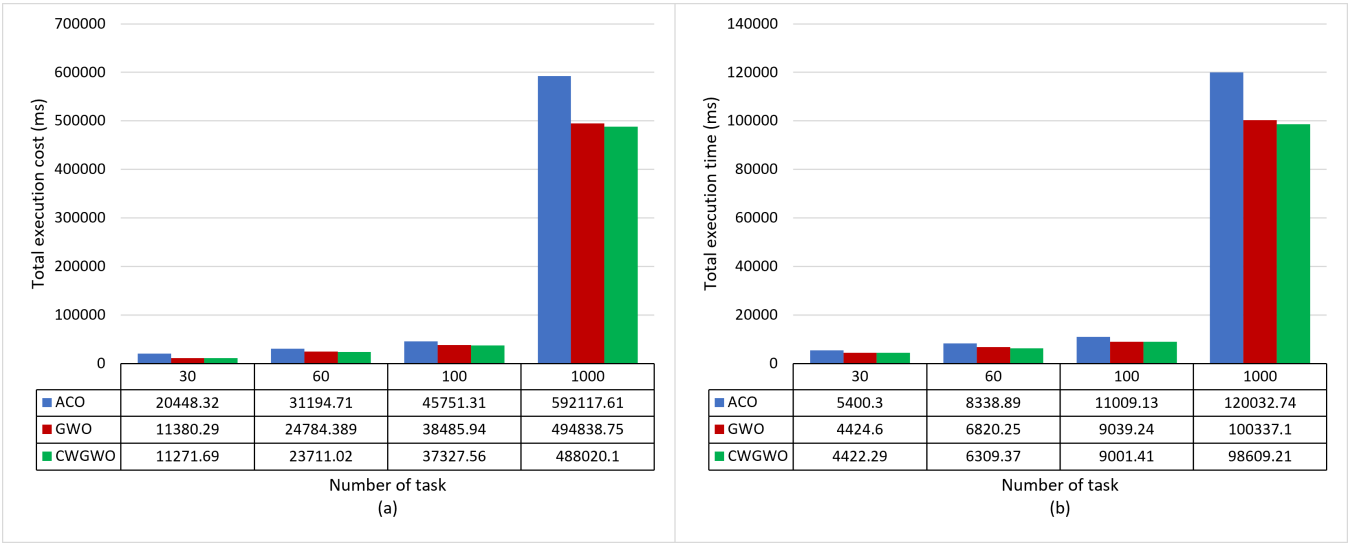


Figure 2: (a) TEC and (b) TET with CyberShake

Figure 3 (a) compares the TEC under the SIPHT workflow with the following task sizes: 30, 60, 100, and 1000. The CWGWO achieves reductions of 79.68%, 25.86%, 18.87%, and 19.65% over ACO, and 0.96%,

4.52%, 3.10%, and 1.39% over GWO for task sizes 30, 60, 100, and 1000, respectively. The reduction is most prominent for small and medium scientific workflows. Figure 3 (b) illustrates TET for SIPHT workflow. CWGWO achieves reductions of 22.05%, 22.2%, 21.79%, and 19.62% over ACO and 0.06%, 8.09%, 1.75%, and 0.21% over GWO for considered tasks, respectively.

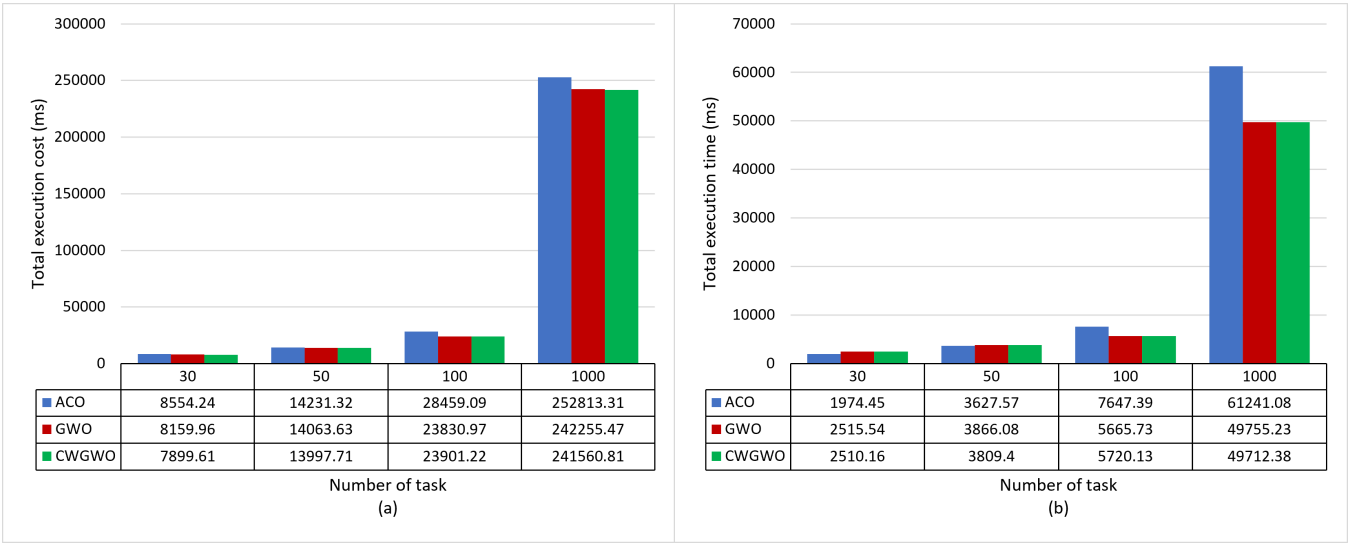


	30	60	100	1000
ACO	20448.32	31194.71	45751.31	592117.61
GWO	11380.29	24784.389	38485.94	494838.75
CWGWO	11271.69	23711.02	37327.56	488020.1

	30	60	100	1000
ACO	5400.3	8338.89	11009.13	120032.74
GWO	4424.6	6820.25	9039.24	100337.1
CWGWO	4422.29	6309.37	9001.41	98609.21

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

Figure 4 (a) represents the TEC under the Inspiral workflow with the following task sizes: 30, 50, 100, and 1000. The proposed CWGWO algorithm reduces TEC by 4.83%, 1.19%, 19.42%, and 4.35% compared to ACO, and by 3.29%, 0.47%, -0.29%, and 0.28% compared to GWO across the given task sizes. Figure 4 (b) shows the TET for the Inspiral workflow. The CWGWO algorithm reduces TET by -21.50%, –6.16%, 34.97%, and 23.08% compared to ACO and by 0.21%, 1.48%, -0.95%, and 0.08% compared to GWO for considered tasks.



	30	50	100	1000
ACO	8554.24	14231.32	28459.09	252813.31
GWO	8159.96	14063.63	23830.97	242255.47
CWGWO	7899.61	13997.71	23901.22	241560.81

	30	50	100	1000
ACO	1974.45	3627.57	7647.39	61241.08
GWO	2515.54	3866.08	5665.73	49755.23
CWGWO	2510.16	3809.4	5720.13	49712.38

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

Figure 5 (a) presents the Montage workflow results. CWGWO reduces TEC by 1.18%, 0.09%, -0.51%, and 0.14% compared to ACO, and 1.44%, 0.36%, 0.20%, and 0.26% compared to GWO for 25, 50, 100, and 1000 tasks, respectively. Improvement is moderate, particularly for large-scale workloads. Figure 5 (b) presents the TEC under the Montage workflow. The CWGWO algorithm shows -20.70%, -11.10%, -4.24%,

and -1.61% reduction compared to ACO and -2.57%, 2.45%, 1.85%, and -0.09% compared to GWO for considered tasks.

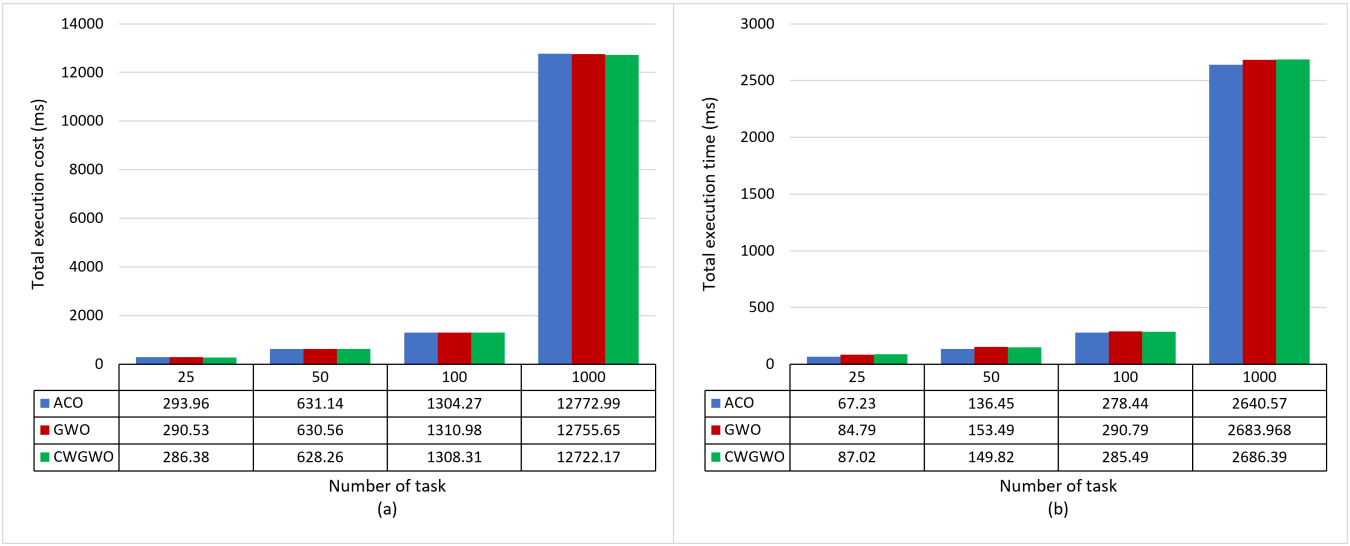


Figure 5: (a) TEC and (b) TET with Montage

Figure 6 (a) illustrates TEC results for the Epigenomics workflow. CWGWO achieves 15.44%, 6.78%, 1.85%, and 24.11% cost reduction over ACO, and 0.55%, 0.09%, 0.08%, and 0.03% over GWO for 24, 47, 100, and 997 tasks. CWGWO consistently performs better than ACO and remains competitive with GWO on large datasets. Figure 6 (b) presents TEC under the Epigenomics workflow. In the Epigenomics workflow, CWGWO achieves -14.37%, -2.24%, -21.38%, and 36.98% improvement compared to ACO and 1.77%, 3.08%, 11.45%, and 2.63% compared to GWO.

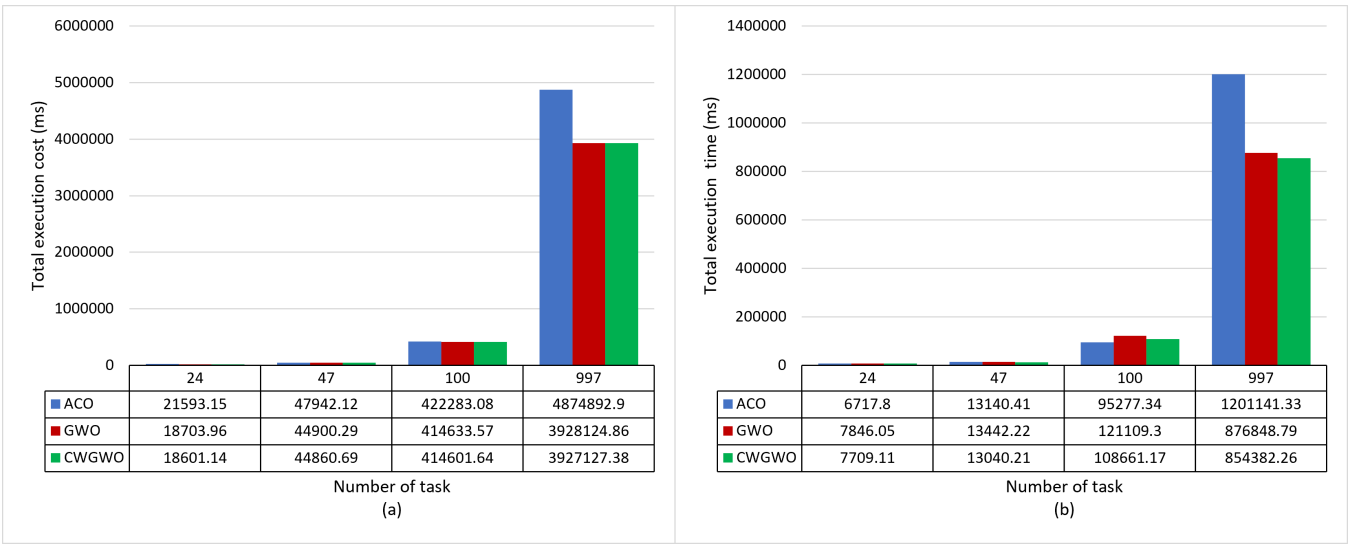


Figure 6: (a) TEC and (b) TET with Epigenomics

4. Conclusion and future direction

The proposed CWGWO 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 various task sizes. The performance

improvement is achieved by using customised weights for the α, β, and δ wolves, rather than equal weighting, in basic GWO, where all leaders are influenced equaly. The CWGWO assigns a weight of 0.5 for α, 0.34 for β, and 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. Figures 2 (a), Figure 3 (a), and Figure 6(a) present the best reductions in TEC, while Figures 2 (b), Figure 3 (b), and figure 4 (b) show the best reductions in TET. For example, TEC decreased by 59.68% for CyberShake (50 tasks), by 79.68% for SIPHT (30 tasks), and by 24.11% for Epigenomics (997 tasks). Similar improvements were observed for TET, including reductions of 52.09% for CyberShake (50 tasks), 22.20% for SIPHT (60 tasks), and 34.97% for Inspiral (100 tasks). These observations confirm that CWGWO provides the best results in many of the tested workflows and maintains performance close to, or better than, GWO in larger workflows.

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.

Declerations

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

Ethics approval and consent to participate: Not applicable.

Availability of data and material: The data will be available based on readers request.

Funding: No specific external funding was received for this study.

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). 1

[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). 1, 1

[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). 1

[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). 1

[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). 1

[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). 1

[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). 1

[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). 1

[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). 1

[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). 1

[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). 1

[13] 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). 1

[14] 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). 1

[15] 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). 1

[16] 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). 1

[17] 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). 1

[18] S. Ijaz, E. U. Munir, S. G. Ahmad, M. M. Rafique, and O. F. Rana, *Energy-makespan optimization of workflow scheduling in fog–cloud computing*, *Computing*, vol. 103, pp. 2033–2059, Springer (2021). 1

[19] 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). 1

[20] 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). 1, 2.1.1, 2.1.2

[21] 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). 1

[22] 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). 1

[23] 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). 1

[24] 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). 1

[25] 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). 1

[26] Chotu Lal, Harish Sharma, and Chetan Sharma, *Workflow Scheduling Using MGWO Algorithm in Cloud Computing*, in *Proceedings of the International Conference on Communication and Intelligent Systems*, pp. 17–28, Springer, 2024. 1

[27] 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). 1

[28] Chotu Lal and Harish Sharma, *A Workflow Scheduling Using an Efficient Hybrid PSO-MGWO Algorithm in Cloud Computing*, *SN Computer Science*, vol. 6, no. 8, p. 929, Springer (2025). 1

[29] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis, *Grey wolf optimizer*, *Advances in Engineering Software*, vol. 69, pp. 46–61, Elsevier (2014). 1

[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). 2.2