THE SCIENTIFIC TEMPER (2026) Vol. 17 (5): 6240-6249

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.5.05

https://scientifictemper.com/

RESEARCH ARTICLE

An Efficient Priority Weight based Modified Firefly Algorithm for Task scheduling in Cloud Computing

N. Kannammal[1*], V. Jude Nirmal[2]

Abstract

In cloud computing, task scheduling poses a significant challenge and is widely adopted by service providers and clients. The primary issue for task schedulers lies in identifying suitable resources (virtual machines) for given tasks while optimizing key metrics such as makespan, cost, energy consumption, and resource utilization. Traditional heuristic approaches often result in suboptimal schedules, resource wastage, or degraded service quality. Meta-heuristic algorithms have gained prominence for tackling task scheduling as a discrete optimization problem. The main objective of this article is to propose a task scheduling algorithm that delivers near-optimal solutions for multi-objective task scheduling in cloud environments. This article proposes an efficient priority weight based modified firefly algorithm (PWMFA) for task scheduling to tackle the problem faced in heuristic approach. The key innovation of this work lies in computing priority weights using a Hamming distance function. The proposed algorithm enhances solution diversity and avoids local optima by leveraging this distance function and priority weighting mechanism. CloudSim was employed to evaluate PWMFA's performance in terms of makespan, total execution time, degree of imbalance, and energy consumption. The results demonstrate that the proposed approach outperforms recent methods by reducing makespan, total execution time, degree of imbalance, and energy consumption.

Keywords: Cloud Computing, Task Scheduling, Meta-Heuristics, Firefly, Multi-objective optimization, Priority weight.

Introduction

The cloud computing offers on-demand, adaptable and expandable service to clients according to their requirements, facilitated by virtualization. It manages enormous data that results in reduced performance if the tasks are not adequately scheduled. Efficient task scheduling directly influences the efficacy of the cloud. Scheduling algorithms are crucial for providing substantial advantages and enhanced performance (Fu, X. et al., 2023). Significant problems in task scheduling encompass flexibility, consistency, load balancing, and dynamic resource allocation. Task scheduling is necessary in the resource management system of cloud computing, ensuring, the attainment of time, cost, energy, resource utilization and others (Saravanan, T., & Ramesh, K. (2024).

The task scheduling approach can be categorized into heuristic, meta-heuristic and hybrid. Heuristic approaches are easy ways to solve problems that rely on experience or rules to obtain better solution, albeit they do not ensure optimality. Min-Min, Max-Min and Round robin are some of the examples of heuristic approach (Ibrahim, I.M. et. al., 2021). Meta-heuristic approaches are sophisticated, high-level tactics that are frequently based on natural phenomena and are intended to efficiently explore and utilize search spaces (Aktan, M. N., & Bulut, H. 2022). Most used meta-heuristic approach includes: Particle swam optimization, Whale optimization, Genetic, and Grey wolf optimization. Hybrid approach integrates meta-heuristic to harness their advantages and enhance performance efficiency (Khan, M.S.A., & Santhosh, R. 2022). Hybrid algorithms in cloud computing combine multiple optimization techniques, such as meta-heuristics with machine learning or swarm

[1]Research Scholar, Department of Computer Science, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India

[2]Assistant Professor & Research Supervisor, Department of Computer Science, St.Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India

***Corresponding Author:** N. Kannammal, Research Scholar, Department of Computer Science, St. Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India, E-mail: kannammal0506@gmail.com

How to cite this article: Kannammal, N., Nirmal, V.J. (2026). An Efficient Priority Weight based Modified Firefly Algorithm for Task scheduling in Cloud Computing. The Scientific Temper, **17**(5):6240-6249.

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.5.05

Source of support: Nil

Conflict of interest: None.

© The Scientific Temper. 2026

Received: 22/03/2026 **Accepted:** 26/04/2026 **Published:** 22/05/2026

intelligence with genetic algorithms, to improve task scheduling, resource allocation, and load balancing. This article employed meta-heuristic approach for task scheduling.

Formulating an optimization model that attains the objectives by identifying the most optimal solution is crucial because of the intrinsic equilibrium among optimization goals (Niu, T., et al., 2023). Consequently meta-heuristic task scheduling encompasses two categories of scheduling objectives: Single and Multi-Objective. The single-objective considers only one specific objective function either minimizes (makespan) or maximizes (resource utilization) the objectives to obtain optimal solution (Banerjee, P., et al., 2023). The multi-objective considers multiple objective functions concurrently. Task scheduling typically represents a multi-objective problem that requires simultaneously optimizing inconsistent goals, such as minimizing makespan and energy consumption while maximizing resource utilization and workload balance (Cui, Z., et al., 2023). This work focus on multi-objective task scheduling to reduce makespan, total execution time, degree of imbalance and energy consumption by considering objective functions.

Multiple authors suggested the multi-objective meta-heuristic based task scheduling (Mangalampalli, S., et al., 2022) (Amer, D. A., et al., 2022) which produce favorable results; nonetheless, they are susceptible to local optima trapping, and their effectiveness is often inadequate. This article proposes an efficient priority weight based modified firefly algorithm (PWMFA) for task scheduling. Firefly algorithm is the one of the category of nature-inspired meta-heuristic algorithm. It provides global optimum solution. This approach improves solution diversity and escapes local optima through effective distance-based prioritization and weighting. It aligns with multi-objective task scheduling needs in cloud environments, extending prior PWMFA evaluations. The suggested algorithm enhances solution variety and circumvents local optima through the utilization of distance function and priority weight. The main contribution of this research is:

- Develop an efficient task scheduling algorithm using modified firefly optimization based on the priority weight.
- Compute the priority weight based on the hamming distance
- Formulate the multi-objective scheduling approach to minimize makespan, total execution time, energy and degree of imbalance.

The subsequent sections of the paper are organized as follows: Section 2 delineates the pertinent literature on diverse task scheduling approaches. Section 3 explains the proposed Efficient Priority Weight based Modified Firefly Algorithm for Task scheduling in Cloud Computing. Section 4 examines the simulation results, while Section 5 presents the conclusion and outlines possibilities for the future work.

Literature Review

Meta-heuristic task scheduling algorithms have gained considerable popularity due to their efficiency in addressing the issues and large scale computing problems. This section explains the different meta-heuristic approach used for schedule tasks. Mangalampalli, S., et al., (2022) suggested Whale optimization algorithm to reduce energy consumption and cost. This approach computes priority values for task and VM and also used multi-objective function for task allocation. Ammari, A.C., et al., (2022) emphasize on delay-bounded applications within distributed datacenters and execute cost and energy efficient scheduling of diverse heterogeneous applications while ensuring compliance with delay bound needs for activities. The challenge of reducing operation cost in datacenters is formulate and effectively optimized using modified firefly algorithm. Nabi, S. et al., (2022) suggested adaptive PSO for task scheduling which decrease execution time, and increase resource utilization. The author introduced adaptive weight approach for improving the local and global search space.

Zhang, H., & Jia, R., (2023) proposed cat swarm optimization for task scheduling. This method optimizes multi-objective task scheduling and attains minimum execution time and cost but it fall into local optimization. Saif, F.A., et al., (2023) suggested grey wolf optimization algorithm to schedule tasks. This approach decreases time and power usage. The results highlight the requirement of improving task allocation to appropriate resource according to task specifications to enhance scheduling performance. Dankolo, N.M., et al., (2024) scheduled tasks using modified flower pollination algorithm. This algorithm combines switch likelihood, inertia weight for levy flight balancing and chaotic map. It minimizes time and maximizes the resource usage.

Zhang, Y., & Wang, J. (2024) suggested enhanced whale optimization algorithm to solve multi-objective scheduling problem. This approach integrates levy flight and utilized crossover method for finding the optimal solution. It holds potential for practical applications, facilitating the optimization of resource management in cloud while addressing different conflicting objectives. Damera, V.K., et al., (2024) employed snake optimization algorithm for task scheduling in cloud. This method utilized sine chaos mapping, spiral search and dynamic adaptive weight to improve the task scheduling performance. Mary, J.M., & Amalarethinam, D.I.G., (2025) suggested modified sunflower optimization algorithm for multi-objective task scheduling. The algorithm enhanced with levy flight, mutation and crossover to optimize exploration and exploitation reduce convergence and local minima. Qasim, M., & Sajid, M., (2025) introduced a scheduling approach based on the firefly algorithm. The modified firefly based scheduler uses transfer function and quantization approach to schedule tasks on VM in a cloud, aiming to reduces the time. It provides low

Table 1: Literature Review Summary

Author & Year	*Methodology*	*Contribution*	*Metrics*	*Limitations / Research Gap*
Mangalampalli, S., et al., (2022)	Priority based Whale optimization	Utilize Whale optimization algorithm to schedule tasks based on task and VM priority.	Cost and energy	High energy usage and slow convergence speed.
Ammari, A.C., et al., (2022)	Firefly algorithm	Employed modified firefly algorithm to improve the convergence speed and global search.	Execution time, cost, energy, bandwidth and convergence speed	Low energy consumption but unable to find appropriate solution
Nabi, S. et al., (2022)	Adaptive PSO	Introduced adaptive inertia weight approach in PSO to adjust search process	Makespan, throughput and average resource utilization ratio	It doesn't optimize the response time and not considered the energy metric
Zhang, H., & Jia, R., (2023)	Cat Swarm Optimization	Employed cat swarm optimization and adjust weight factor and fitness for optimizing task scheduling	Execution time, cost, and optimization effect	It easily suffer to local optimization
Saif, F.A., et al., (2023)	Grey wolf optimization	Multi-objective grey wolf optimization for task scheduling to reduce the time and power usage	Delay and energy consumption	This approach affect resource usage and leads to load imbalance
Dankolo, N.M., et al., (2024)	Modified flower pollination	Modified flower pollination algorithm combined with dynamic switch likelihood, inertia weight and chaotic map	Makespan, Cost and resource utilization	It decreases global search and high computational overhead
Zhang, Y., & Wang, J. (2024)	Enhanced Whale optimization	Enhanced whale optimization algorithm integrated with levy flight and crossover method for multi-objective scheduling	Resource utilization, energy consumption and cost	It reduce reliability and performance which leads to high cost and resource wastage
Damera, V.K., et al., (2024)	Snake Optimization Algorithm	Advanced snake optimization with sine chaos mapping, spiral search and dynamic adaptive weight	Makespan, Energy and Convergence speed	High Computation complexity and doesn't handle large scale system
Mary, J.M., & Amalarethinam, D.I.G., (2025)	Modified Sunflower optimization	Modified sunflower optimization with levy flight, crossover and mutation for efficient task scheduling	Makespan, cost and resource utilization	Doesn't consider energy metric and tested lower number of tasks (500) and VMs (50)
Qasim, M., & Sajid, M., (2025)	Modified Firefly algorithm	Modified firefly algorithm with transfer function and Quantization method	Makespan and convergence speed	High computational complexity

convergence and high result quality. Table 1 shows the summary of related works.

This comparative analysis of 10 meta-heuristic algorithms reveals diverse modifications to Whale, Firefly, PSO, Cat Swarm, Grey Wolf, Flower Pollination, and Sunflower optimizers for cloud task scheduling, emphasizing priority-based, adaptive, levy flight, chaos maps, and inertia weight enhancements to boost convergence, global search, and task-VM allocation. Evaluated primarily on makespan, execution time, cost, energy, resource utilization, throughput, delay, and convergence speed, these approaches show trade-offs in multi-objective performance. Common limitations include high computational complexity, slow convergence, local optima trapping, load imbalance, scalability issues for large datasets, and overlooked metrics like energy or response time.

PWMFA addresses these gaps through Hamming distance-based priority weighting, achieving superior reductions in makespan, imbalance, and energy via CloudSim on 200–1000 tasks and 40–200 VMs.

Proposed Methodology

The Priority Weight-based Modified Firefly Algorithm (PWMFA) enhances task scheduling through Hamming distance-driven prioritization.

Key Parameters

Standard firefly algorithm parameters like light absorption coefficient (γ, typically 1.0), randomization parameter (α, often 0.2), and population size form the base. PWMFA likely extends these with priority weights for tasks and a Hamming distance threshold for prioritization, though exact values depend on implementation specifics.

Task and VM Profiling

Compute priority weights for tasks and VMs using Hamming distance between feature vectors (e.g., length, MIPS, bandwidth), ranking them for initial allocation preference.

Population Initialization

Generate firefly population representing task-VM mappings, with brightness inversely proportional to weighted makespan, energy, and imbalance costs.

Fitness Evaluation

Assess each solution via multi-objective function incorporating priority-weighted makespan, energy consumption, load imbalance degree, and resource utilization.

Firefly Movement with Priority Guidance

Update positions using modified attraction: brighter fireflies pull others with adaptive randomization (Levy flight) and priority-biased step sizes to boost diversity and escape local optima.

Iteration and Termination

Evolve over iterations until convergence or max limit; select global best schedule minimizing all objectives, validated in CloudSim with 200–1000 tasks and 40–200 VMs.

In this work, priority weight modified firefly algorithm was proposed to schedule the tasks in cloud. The proposed PWMFA approach is multi-objective meta-heuristic technique which has derived the fitness function by reducing the makespan, total execution time, degree of imbalance and energy consumption. The tasks are scheduled based on the priority weight which can be computed using hamming distance function. Figure 1 shows the system architecture of proposed PWMFA.

The task scheduling problem can be defined as, follows: A set of tasks T = {t_1,t_2,t_3,…,t_n}are mapped to a set of virtual machines VM = {vm_1,vm_2,vm_3,…vm_m}by using priority weight computed using hamming distance function which reduce multi-objectives such as makespan, total execution time, degree of imbalance and energy consumption. Each task in task set contains two attributes: $task_id_i$ and $task_len_i$. The $task_id_i$ denotes the i^{th} task identification number and $task_len_i$ indicates task length in Million Instruction Per Second (MIPS). Each vm in VM set has there attributes: vm_id_j, vm_ps_j and vm_pcr_j. The vm_id_j is the j^{th} VM identification number. The vm_ps_j indicates the processing speed and vm_pcr_j denotes power consumption rate.

The proposed PWMFA employs Firefly Algorithm (FA) for task scheduling due to its effectiveness in solving NP-hard problems with fewer iterations and extensive exploration of the search space. FA represents a simple yet powerful swarm intelligence technique for tackling complex optimization challenges, inspired by the bioluminescent signaling and mutual attraction of fireflies in nature. Brighter fireflies (those with higher light intensity, corresponding to better objective function values) attract less bright ones, with brightness quantified as follows:

$$\beta_{ij} = \beta_0 \times e^{-\gamma r_{ij}^2} \quad (1)$$

Where β_{ij} represents the brightness (attraction) between two fireflies. r_{ij} denotes the distance between two fireflies,β_0 signifies the initial attraction factor and γ denotes the constant value of the light absorption coefficient. The value $\beta_0$and γ are set as 1. The distance r_{ij} between two fireflies (f_i and f_j) can be computed as,

$$r_{ij} = \sqrt{\sum_{d=1}^{D}(f_{id} - f_{jd})^2} \quad (2)$$

Where D denotes the dimension of the problem, d signifies the d-th dimension of the firefly's location vector. During the search, firefly f_j is drawn to the more brightness firefly f_i and advances towards f_i. The position is updated as follows:

$$f_j(t+1) = f_j(t) + \beta_{ij} \times (f_i(t) - f_j(t) + \alpha \times \varepsilon) \quad (3)$$

Where t indicates the current iteration number, α and ε are arbitrary number between [0, 1] and [-0.5, 0.5].

During the optimization phase, all fireflies will converge towards the brightest firefly, eventually resulting in the aggregation of firefly individuals around the firefly exhibiting the highest brightness of finalize the optimization process.

In this paper, the firefly algorithm is modified with priority weight and hamming distance function. Algorithm-1 explains the proposed PWMFA.

In algorithm-1, the parameters number of firefly population, maximum iteration and parameters related to firefly algorithm are initialized and the firefly population is randomly generated. The firefly population contains N solution. Each firefly population FP_i= {$fp_{i,1}$, $fp_{i,2}$, $fp_{i,3}$,…,$fp_{i,N}$} consists of n tasks with each task denoting the identification of a VM on which the tasks are executed. The following equation is used to randomly generate solutions within the population.

$$fp_{ij} = random(1,m) \quad (4)$$

Consider ten tasks {t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}} and four VMs {vm_1, vm_2, vm_3, vm_4}. A initial solution of FP_i = {4 2 3 1 2 1 2 4 3 2}, it means the tasks {t_4, t_6} allocated to vm_1,tasks {t_2, t_5, t_7, t_{10}} assigned to vm_2, tasks {t_3, t_9} allotted to vm_3 and {t_1,

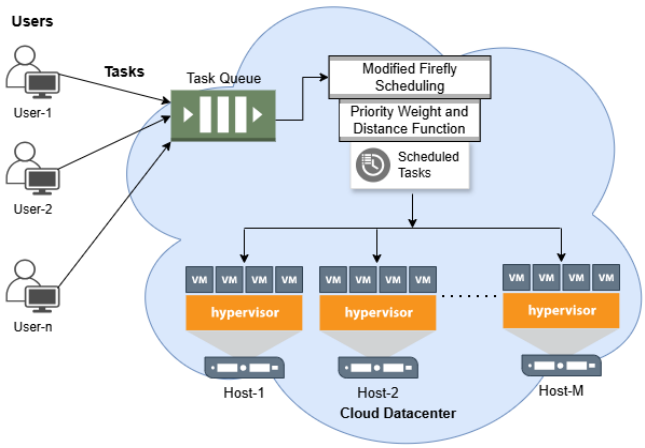


Figure 1: General Architecture of PWMFA

Algorithm 1: Priority weight based Modified Firefly Algorithm

Input: T = {t_1, t_2, …., t_n}, VM= {vm_1, vm_2, …., vm_m}
Output: Allocation of Task to VMs
Step1: Initialize Parameters:
Step2: Randomly generate initial firefly population
Step3: For each pop
Step4: Compute priority weight using Algorithm-2
Step5: Compute fitness value (Fit) using Eq. (6)
Step6: End For
Step7: For it = 1 to MaxIter do
Step8: For i = 1 to pop do
Step9: For j = 1 to pop do
Step10: If Fit (fp_j) < Fit (fp_i) then
Step11: Compute distance between fp_i and fp_j using Eq. (5)
Step12: Compute beta using Eq. (1)
Step13: Modify the fp_i position using Eq. (12)
Step14: Compute fitness value (Fit) for fp_i using Eq. (6)
Step15: EndIf
Step16: EndFor
Step17: EndFor
Step18: Update the population
Step19: EndFor
Step20: return best population

t_8} allocated to vm_4. For each firefly population, computes priority weight based on the hamming distance. Algorithm-2 explains the priority weight computation. The hamming distance between two fireflies can be computed as,

$$hammDist(FP_i, FP_j) = \sum_{k=1}^{n} 1(FP_{ik} \neq FP_{jk}) \quad (5)$$

The hamming distance is calculated for current firefly population with remaining population. The distance values are sorted according to ascending order. Select top 'k' lowest distance from dist[] and find the sum. The priority weight is obtained by dividing the sum by k. The value of k is depends on the number of tasks. After priority weight computation, the fitness value for each firefly population was estimated.

In this article, makespan, total execution time, degree of imbalance and energy consumption are used to compute the fitness value. The formula for fitness computation as,

$$Fit = MK + TET + DoI + TEC \quad (6)$$

Where MK indicates makespan, TET refers to total execution time, DoI denotes degree of imbalance and TEC refers to total energy consumption.

The makespan (MK) is defined as the maximum processing time across all resources. It can be computed as,

$$MK = Max\left(ET_{vm_j}\right) \quad \forall\ 1 \leq j \leq m \quad (7)$$

Algorithm 2: Priority Weight Computation

Input: Population pop[][], CurrentPopulation, CP_i, k
Output: Priority Weight (PW_i)
Step01: Initialize dist[]
Step02: For i = 1 to length (pop)
Step03: Compute dist[i] = *hammDist*(CP_i, pop[i]) using Eq.(5)
Step04: EndFor
Step05: Sort dist by ascending order
Step06: Sum = 0
Step07: For i =1 to *Min*(k, length(dist))
Step08: Sum = Sum + dist [i]
Step09: EndFor
Step10: PW_i = Sum / k
Step11: Return PW_i

Where ET_{vm_j} is denotes the VM execution time of j^{th}VM. It can be calculated as follows:

$$ET_{vm_j} = \sum_{i=0}^{n} \frac{task_len_i}{vm_ps_j} \quad \forall\ 1 \leq j \leq m \quad (8)$$

Where $task_len_i$ and vm_ps_j indicates the length of i^{th} task and processing speed of j^{th} VM respectively.

Total execution time (TET) refers to the complete amount of time taken by a program or process to run from start to finish.
TET is calculated as,

$$TET = \sum_{j=1}^{m} ET_{vm_j} \ \forall\ 1 \leq j \leq m \quad (9)$$

The degree of imbalance (DoI) denotes the extent of uneven workload allocation among various units, such processors, tasks, or resources within a parallel of distributed system. It signifies the extent to which one part of the system performs more work relative to others. It can be computed as,

$$DoI = m * \left(\frac{MK - VM_{minET}}{\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{task_{len_i}}{vm_{ps_j}}}\right) \quad (10)$$

Where n and m indicates number of tasks and VMs. VM_{minET} denotes the minimum execution time of every VM.

The total energy consumption (TEC) is computed using,

$$TEC = \sum_{j=1}^{m} ET_{vm_j} \times vm_pcr_j \quad (11)$$

Where vm_pcr_j indicates the power consumption rate of j^{th} VM

After computing the fitness values, each firefly assesses its brightness (lowest minimum) relative to others and moves towards any firefly exhibiting greater brightness (lowest fitness). The movement is governed by two factors: the attraction, which diminishes with the distance between solutions and a stochastic element to facilitate exploration.

If the fitness value of firefly (fp_j) is less than firefly (fp_i) then, move the firefly fp_i towards firefly fp_j. The following equation was used to update the new position of fp_i.

$$f_i(t+1) = f_i(t) + \beta_{ij} \times \left(f_j(t) - f_i(t) + \alpha \times rand + \delta * PW_i\right) \quad (12)$$

Where t indicates the current iteration number. α and rand represent arbitrary number between [0, 1]. PW_i denotes priority weight of i^{th} population. β_{ij} represents the brightness (attraction) between two fireflies f_i and f_j.

After updating the position of fp_i compute the fitness value using Eq. (6). The new firefly population express a synthesis of the original firefly's traits and those of the more brightness firefly it met. Randomness guarantees diversity, inhibiting premature convergence of the algorithm. The evaluation of fitness computation, the firefly population movement, and the generation of the new solution is repeated for multiple iterations. Figure 2 shows the workflow of the proposed PWMFA.

The priority-weight–modified firefly algorithm effectively optimizes task scheduling by assigning weights and updating task positions so that higher-weight tasks are mapped to suitable resources, thereby minimizing makespan and total execution time, improving load balance by reducing the degree of imbalance across virtual machines, and lowering overall energy consumption through shorter, more evenly distributed execution.

The priority weight based modified firefly algorithm efficiently schedule tasks to resources. It compute priority weight and fitness value for each population. The proposed firefly algorithm updates the highest weight population and provides optimal allocation results which effectively reduces makespan, DoI and energy consumption. The proposed approach efficiently handle multi objective problem, improve convergence speed and provide strong global search capability.

Result and Discussion

This section evaluates the performance of the proposed PWMFA approach using cloudsim, a toolkit specifically developed for modeling cloud computing infrastructures. The randomly generated tasks are utilized to evaluate the efficiency of the proposed approach. The tasks ranges from 200 to 1000 and VM from 40 to 200 utilized for simulation. The efficiency of the proposed PWMFA was compared with PSO Alsaidy, S. A., et al., (2022) and traditional firefly algorithms in terms of makespan, total execution time, degree of imbalance and energy consumption. Table 2 shows the simulation configuration.

The performance evaluations are based on the two diverse set of simulated conditions. In first settings, the number of tasks varies from 200 to 1000 and the number of VM is fixed as 100. In the next settings, the VM varies from 40 to 200 and the task count is fixed as 500.

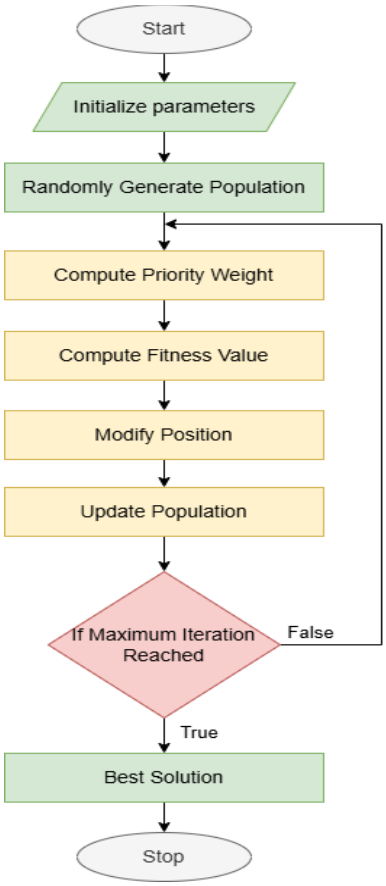


Figure 2: Workflow of PWMFA

Table 3 and Figure 3 shows the performance comparison of makespan for fixed VM and varying task count. The results indicate that an increase in task count correlates with an increase in makespan. The proposed approach achieves makespan of 3.15, 5.61, 8.59, 10.48, and 11.65 for 200, 400, 600, 800 and 1000 tasks respectively.

The proposed PWMFA approach has lowest makespan compared to other approaches. The PSO has the larger

Table 2: Simulation Parameter

Parameters	*Values*
Task Count	200 – 1000
VM Count	40 – 200
Task Length (MI)	1000 – 4000
VM MIPS	1000 – 5000
VM PCR (Watt)	200 – 1000
Population Count	10
Maximum Iteration	150
β0	1
α	0.4
γ	1
δ	0.2

Table 3: Makespan Comparison (Fixed VM and Different Task Count)

Number of Tasks	*PSO*	*FA*	*PWMFA*
200	4.0	3.82	3.15
400	7.5	7.1	5.61
600	10.4	9.78	8.59
800	15.2	13.4	10.48
1000	20.7	15.65	11.65

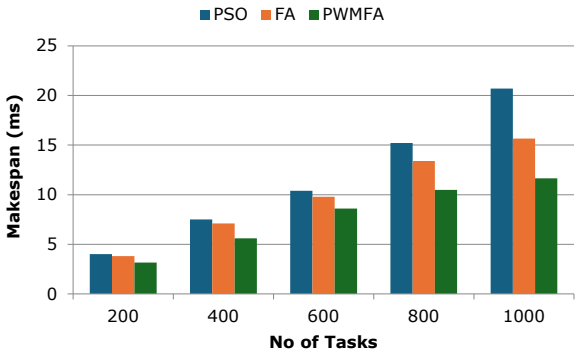


Figure 3: Comparison of Makespan for fixed VM and various tasks

makespan values across all diverse tasks. From figure 3, the proposed PWMFA decreases the makespan (1000 tasks and 100 VMs) by 43.71% for PSO and 25.5% for FA.

Table 4 and Figure 4 shows the performance comparison of total execution time for fixed VM and varying task count. The results indicate that an increase in task count correlates with an increase in total execution time. The proposed approach achieves total execution time of 108.25, 213.71, 430.46, 542.02, and 714.38 for 200, 400, 600, 800 and 1000 tasks respectively.

The proposed PWMFA approach has lowest total execution time compared to other approaches. The PSO has the larger total execution time across all diverse tasks. From Figure 4, the proposed PWMFA decreases the total execution time (1000 tasks and 100 VMs) by 23.27% for PSO and 9.91% for FA.

Table 5 and Figure 5 displays the performance of DoI comparison for fixed VM and various task count. The proposed approach attains DoI of 2.1, 1.93, 1.71, 1.5, and 1.3 for 200, 400, 600, 800 and 1000 tasks respectively.

Table 4: Total Execution Time Comparison (Fixed VM and Different Task Count)

Number of Tasks	*PSO*	*FA*	*PWMFA*
200	155.9	121.70	108.25
400	328.4	279.33	213.71
600	520.9	497.41	430.46
800	690.4	613.42	542.02
1000	931.1	792.97	714.38

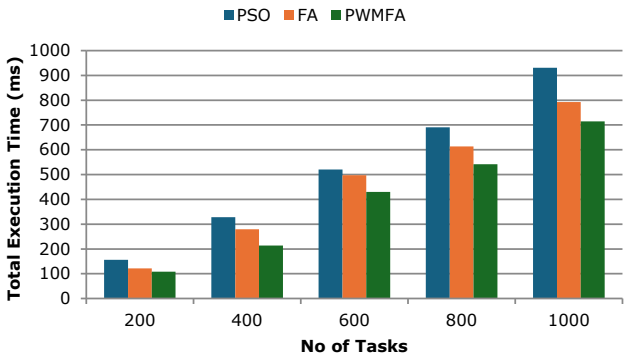


Figure 4: Comparison of Total Execution Time for fixed VM and various tasks

The proposed approach has lowest DoI compared to other approaches. From Figure 5, the proposed PWMFA reduces DoI (1000 tasks and 100 VMs) by 27.7% for PSO and 31.57% for FA.

Table 6 and Figure 6 shows the performance of energy consumption comparison for fixed VM and various task count. The proposed approach consumes energy of 70, 153, 255, 267 and 390 for 200, 400, 600, 800 and 1000 tasks respectively.

From Figure 6, the proposed PWMFA reduces energy consumption (1000 tasks and 100 VMs) by 58.28% for PSO and 50% for FA. The proposed PWMFA has lowest energy consumption compared to PSO and FA.

Table 7 and Figure 7 shows the comparison of makespan of fixed task and diverse VM count. When increasing the number of VMs, the makespan is reduced. The suggested method has makespan of 12.7, 9.93, 5.84, 4.87 and 3.2 for 40, 80, 120, 160, and 200 VMs.

Table 5: DoI Comparison (Fixed VM and Different Task Count)

Number of tasks	*PSO*	*FA*	*PWMFA*
200	2.4	2.9	2.1
400	2.0	2.4	1.93
600	1.8	2.1	1.71
800	1.7	1.9	1.5
1000	1.8	1.9	1.3

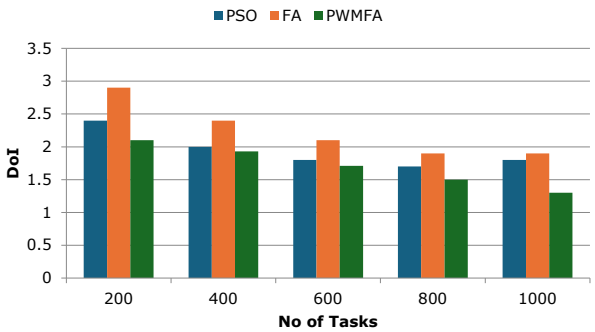


Figure 5: Comparison of DoI for fixed VM and various tasks

Table 6: Energy Consumption Comparison (Fixed VM and Different Task Count)

Number of Tasks	*PSO*	*FA*	*PWMFA*
200	189	150	70
400	368	298	153
600	514	421	255
800	685	590	267
1000	935	780	390

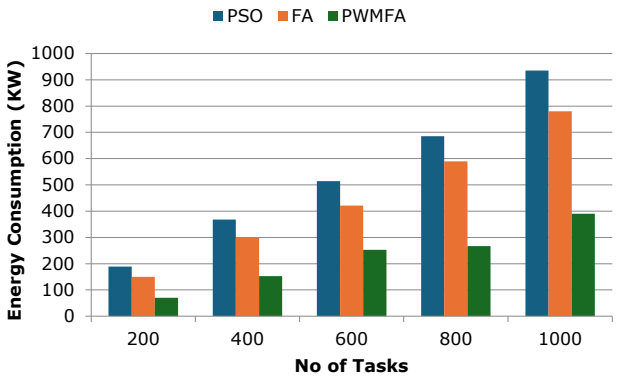


Figure 6: Comparison of Energy Consumption for fixed VM and various tasks

Table 7: Makespan Comparison (Fixed Task and Different VM Count)

Number of VMs	*PSO*	*FA*	*PWMFA*
40	25.1	20.5	12.7
80	14.9	11.3	9.93
120	9.3	8.9	5.84
160	7.1	8.1	4.87
200	7.2	7.5	3.2

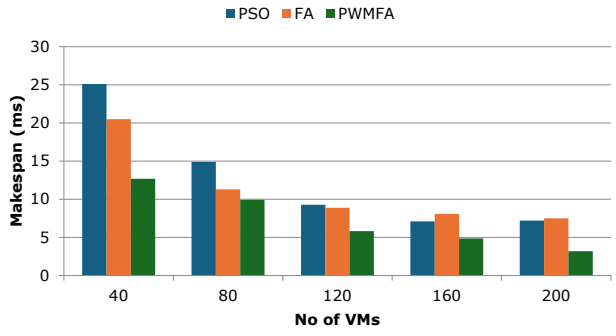


Figure 7: Comparison of Makespan for fixed task and various VMs

The suggested PWMFA reduces the makespan compared to other algorithms. From figure 7, the PWMFA decreases the makespan (500 tasks and 200 VMs)for PSO by 55.5%, and FA by 57.3%.

Table 8 and Figure 8 shows the comparison of total execution time of fixed task and diverse VM count. When

Table 8: Total Execution Time Comparison (Fixed Task and Different VM Count)

Number of VMs	*PSO*	*FA*	*PWMFA*
40	475.6	386.2	272.1
80	526.6	356.4	265.4
120	436.5	311.7	258.9
160	415.8	293.5	239.1
200	443.8	278.1	201.6

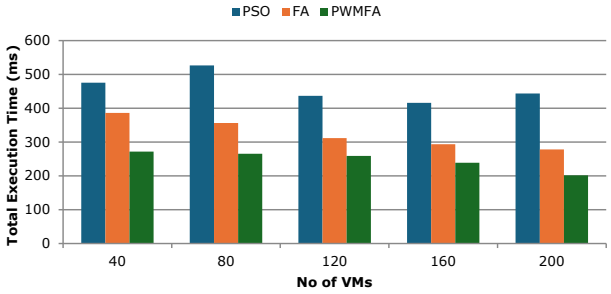


Figure 8: Comparison of Total Execution Time for fixed task and various VMs

increasing the number of VMs, the total execution time is reduced. The suggested method has total execution time of 272.1, 265.4, 258.9, 239.1 and 201.6 for 40, 80, 120, 160, and 200 VMs.

The suggested PWMFA reduces the total execution time compared to other algorithms. From figure 8, the PWMFA decreases the total execution time (500 tasks and 200 VMs) for PSO by 54.5%, and FA by 27.5%.

Table 9 and Figure 9 display the comparison of DoI for fixed task and diverse VM count. The suggested method has DoI of 1.1, 1.5, 1.7, 1.9 and 2.0 for 40, 80, 120, 160, and 200 VMs.

The PWMFA decreases the DoI compared to other algorithms. From figure 9, the PWMFA reduces the DoI (500 tasks and 200 VMs)for PSO by 25.9%and FA by 29.32%.

Table 10 and Figure 10 display the energy consumption comparison for fixed tasks and diverse VM count. The proposed PWMFA attains energy consumption of 185, 213, 202, 198, and 194 for 40, 80, 120, 160 and 200 VMs.

The PWMFA reduces the energy consumption compared to other algorithms. From figure 10, the PWMFA reduces the energy (500 tasks and 200 VMs)for PSO by 67.5%and FA by 7.6%.

Table 11 shows the performance metrics comparison of different methods. Diverse approaches may use different metrics for comparison. The approach EWOA (Zhang, Y., & Wang, J., 2024) concentrates on makespan, total execution time and DoI. The method MSO (Mary, J.M., & Amalarethinam, D.I.G., 2025) used makespan, cost and resource utilization for comparison. The method ESFO (Emami, H., 2022) only considers makespan and energy consumption. Similarly HGWW (Ababneh, J., 2021) not concentrated on DoI metric.

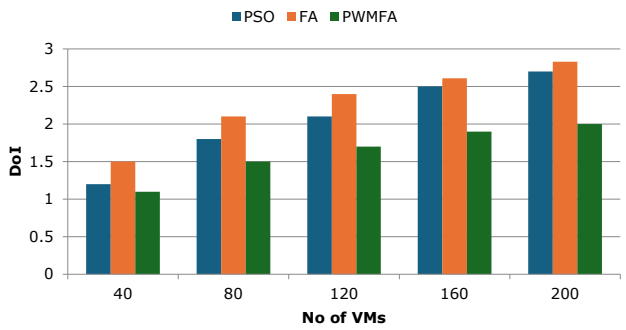


Figure 9: Comparison of DoI for fixed Task and various VMs

Table 9: DoI Comparison (Fixed Task and Different VM Count)

Number of VMs	*PSO*	*FA*	*PWMFA*
40	1.2	1.5	1.1
80	1.8	2.1	1.5
120	2.1	2.4	1.7
160	2.5	2.61	1.9
200	2.7	2.83	2.0

Table 10: Energy Consumption Comparison (Fixed Task and Different VM Count)

Number of VMs	*PSO*	*FA*	*PWMFA*
40	393	220	185
80	495	270	213
120	460	250	202
160	522	239	198
200	598	210	194

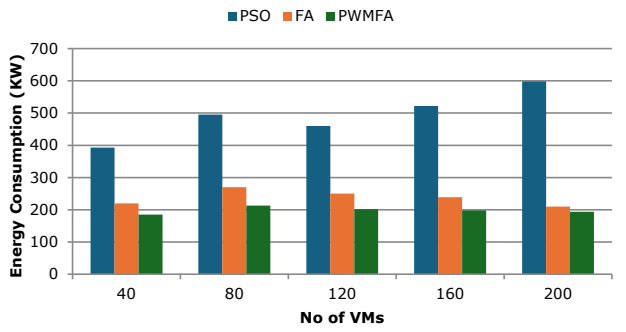


Figure 10: Comparison of Energy Consumption for fixed task and various VMs

For tasks (200, 400,600,800,100) on 100 VMs tested for PSO, FA, and PWMFA algorithms. The metrics makespan, total execution time, DoI, and energy consumption shows more proficient for PWMFA compared to other approaches. The PWMFA reduces average makespan by 31.69% over PSO and 20.64% over FA. The total execution time is reduced by 23.52% and 12.84% for PSO and FA. The proposed PWMFA reduces DoI by 11.95% compared to PSO and 23.75% compared to FA. The PWMFA reduces average energy consumption by 57.82% over PSO and 49.30% over FA.

Table 11: Performance metrics comparison

Methods	*Makespan*	*TET*	*DoI*	*TEC*
EWOA	-	-	-	125
MSO	5.73	-	-	-
ESFO	58	-	-	20
HGWW	100	-	0.4	250
PWMFA	3.15	108.25	2.1	70

Similarly, for VMs (40, 80, 120, 160, 200) on 500 tasks tested for PSO, FA, and PWMFA algorithms. The makespan is reduced by 42.54% and 35.09% for PSO and FA. The PWMFA reduces average total execution time by 46.17% over PSO and 23.91% over FA. The proposed PWMFA reduces DoI by 20.38% compared to PSO and 28.32% compared to FA. The PWMFA reduces average energy consumption by 59.80% over PSO and 16.56% over FA. Overall, the proposed PWMFA algorithm is better than other methods in all metrics.

Meta-heuristic algorithm can provides global optimum solution for task scheduling problem. This work uses nature inspired based firefly algorithm to schedule tasks in cloudThe overall time complexity is O(n3).

Conclusion and Future Work

The increasing demand for the cloud services requires effective, task scheduling algorithms to manage task in cloud environment. Meta-heuristic algorithm can provides global optimum solution for task scheduling problem. This work uses nature inspired based firefly algorithm to schedule tasks in cloud.This paper present an efficient priority weight based modified firefly algorithm for task scheduling. This work computes the priority weight based on hamming distance function. The PWMFA enhances solution variety and circumvents local optima through the utilization of distance function and priority weight.The assessment of PWMFA in comparison to existing algorithms demonstrated its efficient in reducing makespan, total execution time, DoI and energy consumption. The results highlighted the capability of PWMFA to tackle the complex issues of multi-objective task scheduling. The PWMFA presents potential for practical applications, facilitating the optimization of resource management in cloud while addressing numerous conflicting objectives. The proposed approach reduces the average of 31.74% and 20.70% of makespan, 12.37% and 24.10% of DoI and 57.82% and 49.21% of energy consumption compared to PSO and FA. The future scope includes: To employ hybrid optimization algorithm for improving the task scheduling efficiency, use real-world data for simulation and include other performance metrics like cost and resource utilization.

References

Ababneh, J. (2021). A hybrid approach based on grey wolf and whale optimization algorithms for solving cloud task scheduling problem. Mathematical Problems in Engineering, 2021(1), 3517145.

Aktan, M. N., & Bulut, H. (2022). Metaheuristic task scheduling algorithms for cloud computing environments. Concurrency and Computation: Practice and Experience, 34(9), e6513.

Amer, D. A., Attiya, G., Zeidan, I., & Nasr, A. A. (2022). Elite learning Harris hawks optimizer for multi-objective task scheduling in cloud computing. The Journal of Supercomputing, 78(2), 2793-2818.

Ammari, A. C., Labidi, W., Mnif, F., Yuan, H., Zhou, M., &Sarrab, M. (2022). Firefly algorithm and learning-based geographical task scheduling for operational cost minimization in distributed green data centers. Neurocomputing, 490, 146-162.

Alsaidy, S. A., Abbood, A. D., & Sahib, M. A. (2022). Heuristic initialization of PSO task scheduling algorithm in cloud computing. Journal of King Saud University-Computer and Information Sciences, 34(6), 2370-2382.

Banerjee, P., Roy, S., Sinha, A., Hassan, M.M., Burje, S., Agrawal, A., Bairagi, A.K., Alshathri, S. and El-Shafai, W., (2023). MTD-DHJS: makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction. IEEE Access, 11, 105578-105618.

Cui, Z., Zhao, T., Wu, L., Qin, A. K., & Li, J. (2023). Multi-objective cloud task scheduling optimization based on evolutionary multi-factor algorithm. IEEE Transactions on Cloud Computing, 11(4), 3685-3699.

Damera, V. K., Vanitha, G., Indira, B., Sirisha, G., &Vatambeti, R. (2024). Improved snake optimization-based task scheduling in cloud computing. Computing, 106(10), 3353-3385.

Dankolo, N. M., Radzi, N. H. M., Mustaffa, N. H., Osman, N. A., Gabi, D., & Yusuf, M. N. (2024). Enhanced task scheduling and resource allocation in edge-cloud continuum using modified flower pollination algorithm. IEEE Access, 12, 162299-162310.

Emami, H. (2022). Cloud task scheduling using enhanced sunflower optimization algorithm. Ict Express, 8(1), 97-100.

Fu, X., Sun, Y., Wang, H., & Li, H. (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. Cluster Computing, 26(5), 2479-2488.

Ibrahim, I.M., Radie, A.H., Jacksi, K., Zeebaree, S.R., Shukur, H.M., Rashid, Z.N., Sadeeq, M.A. and Yasin, H.M., (2021). Task scheduling algorithms in cloud computing: A review. Turkish Journal of Computer and Mathematics Education, 12(4), 1041-1053.

Khan, M. S. A., & Santhosh, R. (2022). Task scheduling in cloud computing using hybrid optimization algorithm. Soft computing, 26(23), 13069-13079.

Mangalampalli, S., Swain, S. K., &Mangalampalli, V. K. (2022). Multi objective task scheduling in cloud computing using cat swarm optimization algorithm. Arabian journal for science and engineering, 47(2), 1821-1830.

Mangalampalli, S., Swain, S. K., &Mangalampalli, V. K. (2022). Prioritized energy efficient task scheduling algorithm in cloud computing using whale optimization algorithm. Wireless Personal Communications, 126(3), 2231-2247.

Mary, J. M., & Amalarethinam, D. I. G. (2025). Modified Sunflower Optimization Algorithm for Task Scheduling in Cloud Computing. Indian Journal of Science and Technology, 18(17), 1365-1376.

Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: adaptive PSO-based task scheduling approach for cloud computing. Sensors, 22(3), 920.

Niu, T., Lei, B., Guo, L., Fang, S., Li, Q., Gao, B., Yang, L. and Gao, K., (2023). A review of optimization studies for system appointment scheduling. Axioms, 13(1)

Qasim, M., & Sajid, M. (2025). An efficient IoT task scheduling algorithm in cloud environment using modified Firefly algorithm. International Journal of Information Technology, 17(1), 179-188.

Saif, F. A., Latip, R., Hanapi, Z. M., & Shafinah, K. (2023). Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing. IEEE Access, 11, 20635-20646.

Saravanan, T., & Ramesh, K. (2024). A Bio-inspired Energy Efficient Dynamic Task Scheduling (BEDTS) scheme and classification for virtualization CDC. Journal of Engineering Research, 12(3), 387-396.

Zhang, H., & Jia, R. (2023). Application of chaotic cat swarm optimization in cloud computing multi objective task scheduling. IEEE Access, 11, 95443-95454.

Zhang, Y., & Wang, J. (2024). Enhanced Whale Optimization Algorithm for task scheduling in cloud computing environments. Journal of Engineering and Applied Science, 71(1), 121.