International Journal of Computer Engineering in Research Trends
ISSN: 2349-7084/ https://doi.org/10.22362/ijcert/2025/v12/i9/v12i901
Volume 12, Issue 9, September 2025
© 2025, IJCERT All Rights Reserved

Research Paper

A Scalable Hybrid Optimized Firefly Optimization with Simulated Annealing for Scheduling Tasks in Cloud Environments

[1*] Stefy Mathew, [2] Mahaveer Kumar Sain

[1] *Research Scholar, Faculty of Computer Science & Informatics, Maharishi Arvind University, Jaipur, Rajasthan, India, Email Id: stefymathew90@gmail.com*
[2] *Professor, Faculty of Computer Science & Informatics, Maharishi Arvind University, Jaipur, Rajasthan, India, Email Id: mahaveersain@gmail.com*

**Corresponding Author(s): mahaveersain@gmail.com*

Received: 05/06/2025, Revised: 07/07/2025, Accepted: 11/09/2025 Published: 30/09/2025

Abstract: Task scheduling is crucial to the performance of the whole cloud computing infrastructure and is thus the most essential need in a cloud computing environment. The process of allocating the most appropriate resources for work while taking various factors into account is known as task scheduling in cloud computing. There is a discrete optimization problem called NP-hard that describes the task scheduling problem. Hybrid swarm optimization is the proposed solution to this issue. Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA) is a new hybrid optimization methodology for cloud computing job scheduling that is scalable. By integrating the exploration capabilities of firefly swarm optimization with the exploitation strengths of simulated annealing, the proposed approach efficiently balances the computational load across virtual machines. The suggested technique was tested through simulations run on the CloudSim simulator. Experimental evaluations were conducted under both dynamic and static load balancing scenarios in terms of performance metrics, including Makespan, Execution Time, Completion Time, Variance, Total Execution Cost, Average and Resource Utilization on 5 to 25 VMs and 100 to 600 number of tasks. The experimental results demonstrate the effectiveness of HOFO-SA, showing a 32.5% reduction in makespan, a 28.7% improvement in execution time, and a 36.8% increase in resource utilization efficiency ratio showed substantial improvements, reaching 0.99 in HOFO-SA compared to 0.84 in compare Firefly Swarm Optimization Task scheduling (FSOTS) traditional approach. These improvements lead to better workload distribution, enhanced system reliability, and reduced processing delays, making HOFO-SA a promising solution for cloud resource scheduling. This work highlights the significance of hybrid optimization techniques in addressing complex scheduling challenges and improving overall cloud computing performance.

Keywords: Cloud computing, Task scheduling, load balancing, NP-hard problem, Optimization algorithms, Firefly Optimization, Simulated Annealing

---

1. Introduction

The development of cloud computing has been a game-changer for distributed computing. Among the many potential uses for cloud computing are data storage, analytics, and IoT apps [1]. The conventional methods by which businesses and individuals deploy services have been revolutionized by cloud computing. Customers can avoid investing in costly computer equipment by taking use of the various online services it offers to registered customers. The characteristics of cloud computing encourage users to switch from traditional platforms to cloud platforms for their operations. The processing capability of cloud computing is incredibly strong when compared to traditional systems. Resource requests are handled as tasks in the cloud, and the right resources are assigned based on

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

user requirements[2]. Scheduling tasks in the cloud is highly dependent on processing costs and resource utilization. Several optimum task-scheduling algorithms make effective use of these factors to guarantee the best possible job completion[3].

Task computation and resource allocation are scheduled within predetermined restrictions as part of the task-scheduling process. Certain optimization goals are used in this procedure to define how cloud computing resources are allocated for task computation mapping [4]. It is critical to select an appropriate task scheduling algorithm because task scheduling takes into account the physical characteristics of resources and the attributes of task execution; this guarantees that tasks are completed on time within a cost-effective framework, which optimizes cloud computing utilization while preserving QoS requirements. The principal computing components used for processing tasks are virtual machines. Whatever their degree of homogeneity or heterogeneity, all virtual machines have some defining features [5]. The task-specific virtual machine is selected using the scheduling method[6]. Researchers choose scheduling methods with the goal of minimizing makespan value and task execution time [7][8].

The VMblueprint specifies that a physical machine (PM) be involved in processing certain data points. The cloud environment employs static load-balancing techniques in response to workload fluctuations. Nevertheless, cloud computing systems are characterized by dynamic workload fluctuations, making static algorithms inadequate. There must be dynamic mechanisms in place for workload distribution among VMs [9]. The unpredictable and diverse character of tasks makes scheduling an NP-hard issue in the CC. Although dynamic scheduling techniques improve efficiency, these pose difficulties for service providers due to increased communication overhead [10][11]. There are a number of methods for handling task scheduling in CC. However, these algorithms are not appropriate for large-scale scheduling issues. The effectiveness of meta heuristic algorithms in finding optimum solutions in polynomial time* has led to their increased attention in recent years. These algorithms include ACO, PSO, GA, cuckoo search, whale optimization algorithm, symbiotic organisms search, and Harris Hawks optimization[12][13]. Problems with premature convergence plague some of these methods, making it impossible to go beyond local minima in big solution spaces and leading to suboptimal solutions that have an effect on system performance and QoS assurances [14].

1.1 Motivation /Novelty and Contribution

The necessity for optimal resource utilization, resource heterogeneity, and changing workloads make cloud computing task scheduling a crucial concern. Traditional metaheuristic algorithms like FSOTS often suffer from premature convergence, local optima stagnation, and inefficient task allocation. To address these limitations, this study proposes a Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA) approach, which integrates the exploration capability of Firefly Optimization Algorithm (FOA) with the exploitation refinement of Simulated Annealing (SA). The novelty of this work lies in the adaptive hybridization mechanism, which enhances convergence speed, prevents local minima trapping, and ensures better load balancing, making it more effective than conventional scheduling methods. This research contributes to improving execution efficiency, cost-effectiveness, and scalability in cloud environments, making HOFO-SA a promising solution for real-world cloud service optimization.

These are the contributions of the study:

- The study introduces a novel hybridization of Firefly Optimization Algorithm (FOA) with Simulated Annealing (SA) to enhance task scheduling efficiency in cloud computing, addressing local optima stagnation issues and improving solution quality.
- The proposed HOFO-SA method adapts to varying workloads and dynamic cloud environments by efficiently balancing task allocation across heterogeneous virtual machines, thereby minimizing makespan and execution costs.
- The model optimizes task distribution by leveraging the exploration-exploitation balance of FOA and the probabilistic refinement of SA, leading to better load balancing and improved cloud resource utilization.
- The methodology is validated against conventional scheduling techniques, demonstrating superior performance in terms of execution time, cost efficiency, and task completion rates.
- The HOFO-SA model is scalable for real-world cloud service providers and integrates with cloud simulators such as CloudSim, ensuring feasibility in practical cloud computing environments.

1.2 Structure of the paper

The following is a table of contents for this research paper: A review of relevant research on CC task scheduling is provided in Section II. The suggested method for

optimizing task scheduling is presented in Section III. This method makes use of a HOFO-SA. The results of the simulations and the evaluation of the performance of the suggested strategy are covered in Section IV. Section V concludes the article by summarizing its main points and outlining possible directions for further study.

2 Literature Review

Since task scheduling has such a profound impact on system performance, it presents a formidable obstacle in the field of CC. It is critical to create an effective algorithm to enhance the task-scheduling process. The next part explores some of the most modern optimization algorithms that are now available for cloud computing task scheduling.

Priya et al. (2024) proposed two algorithms Improved Whale Optimization Algorithm over Ant Colony Algorithm. Each group has 120 samples with 80% G power, 0.05% threshold, and 95% CI. Task Scheduling Time (ms) is returned for inputs. The findings come from IBM SPSS. Results indicate a significant independent sample t-test ($p= 0.000$, $p<0.05$). The algorithms vary statistically. Ant Colony takes longer to schedule tasks than Improved Whale Optimization Algorithm. According to study, the Improved Whale Optimization Algorithm (53.6000 MS) has a shorter Task Scheduling Time than the Ant Colony Algorithm (93.1000 MS)[15].

Ganesan et al. (2024) combination of Resource Allocation and Task Scheduling using Improved Bat Optimization Algorithm (IBOA) by using dynamic weights with the speed update formula and increasing the random searchability and Modified Social Group Optimization (MSGO) by modifying the acquiring phase is introduced in this research. For 100 tasks, the proposed IB and MSGO methods achieved 32.5 watts of energy. The obtained results exposed that IB and MSGO method has better performance compared with the existing method namely MOTSGWO in terms of energy consumption[16].

Priyadarshini et al. (2024) provide an enhanced DTO-CNN approach to better allocate resources and schedule tasks in a cloud environment. The functioning of the proposed DTO-CNN algorithm is evaluated using throughput, reliability and response time. The developed DTO-CNN has obtained throughput of 307ms, reliability of 0.895 and response time of 991ms for 500 tasks compared to the existing FFARS-MSO model[17].

Zhang et al. (2023) the difficulty of scheduling tasks with several objectives led to the introduction of the cat swarm optimization model. The suggested model has a 6.23-second task execution time and tends to converge after 102 iterations. A total of 6.51 seconds is required to complete a task using the conventional optimization methodology. A total of 6.96 seconds are required to complete a task using the particle swarm optimization model. This suggested approach outperforms the other two in terms of task execution cost, coming in at 3326 yuan when the number of tasks is 500[18].

Liu et al. (2023) implement the GWO algorithm's elite representative exploration mechanism into the population's global search phase to enhance the algorithm's global search capability and prevent it from entering a local optimum state. Simulation results demonstrate that the suggested algorithm outperforms the comparison algorithm for scheduling tasks with varying distributions and sizes, lowering makespan and increasing average resource utilization rate and clearly outperforming algorithm convergence and convergence accuracy[19].

Tian et al. (2023) a particle swarm optimization technique (AWPSO) based on an adaptive weight approach is suggested. The method is repeatedly optimized to enhance the particle swarm scheduling algorithm's capacity for self-optimization by managing the learning factors, speed growth, and other parameters in real time. Results from experiments on the CloudSim platform demonstrate that compared to pre-optimization, the AWPSO algorithm's particle learning ability is both quicker and more accurate [20].

Srichandan et al. (2023) primary purpose of this study is to investigate the effectiveness of the multi-objective Cat Swarm Optimization (TA-CSO) in optimizing job scheduling for improved efficiency. The outcome achieved through the utilization of TA-CSO is likewise replicated by the implementation of an open-source cloud platform known as CloudSim. The research data investigated through the study demonstrates that the proposed TA-CSO scheduling approach delivers the optimal blend of performance results among different objectives[21].

Below Table 1 summarizes methodologies, findings, advantages, limitations, and possible future directions for each study for task scheduling in cloud computing using various optimization algorithms.

Table 1. Summary of the Related Work for Task Scheduling in Cloud Computing Using Various Optimization Algorithms

Reference	Methodology	Results Analysis	Advantages	Limitations	Future Work
Priya et al. (2024)	Improved Whale Optimization Algorithm vs. Ant Colony Algorithm	Improved Whale Optimization Algorithm (53.6ms) outperforms Ant Colony (93.1ms) in task scheduling	Faster scheduling time, statistically significant improvement	Limited dataset (120 samples), lacks real-world validation	Extend to large-scale cloud computing environments
Ganesan et al. (2024)	Improved Bat Optimization Algorithm (IBOA) + Modified Social Group Optimization (MSGO)	IB-MSGO achieves 32.5W energy consumption, better than Multi-Objective Task Scheduling Grey Wolf Optimization (MOTSGWO)	Reduced energy consumption, dynamic weight update	Limited tasks (100), not tested for large-scale scheduling	Implement in heterogeneous cloud infrastructure
Priyadarshini et al. (2024)	Improved Dipper Throated Optimization with Convolutional Neural Network (DTO-CNN)	Throughput: 307ms, Reliability: 0.895, Response Time: 991ms (for 500 tasks)	High reliability and throughput, effective for cloud environments	High response time compared to some other methods	Optimize response time, test scalability
Zhang et al. (2023)	Multi-objective task scheduling using Cat Swarm Optimization (CSO)	Execution time: 6.23s (better than traditional: 6.51s, PSO: 6.96s), Cost: 3326 yuan (500 tasks)	Faster execution, cost reduction	May not generalize well to varying workloads	Apply to real-world cloud platforms
Liu et al. (2023)	Grey Wolf Optimization (GWO) with elite representative exploration mechanism	Reduced makespan, higher resource utilization, improved convergence	Enhanced global search, avoids local optima	Computational complexity	Implement hybrid models for optimization
Tian et al. (2023)	Adaptive Weight Strategy-based Particle Swarm Optimization (AWPSO)	Faster and more accurate scheduling in CloudSim	Real-time adaptability, self-optimizing algorithm	Requires fine-tuning of parameters	Extend to real-world cloud systems
Srichandan et al. (2023)	Multi-objective Cat Swarm Optimization (TA-CSO) in CloudSim	Optimal scheduling balance across objectives	Effective balance in job scheduling	Performance depends on cloud platform parameters	Apply in distributed cloud environments

3. Methodology

The proposed solution, named HOFO-SA, stands for Scalable Hybrid Optimized Firefly Optimization with Simulated Annealing to handle task scheduling operations in cloud computing environments. The integrated Firefly Optimization-Simulated Annealing method functions to optimize dynamic loading processes by handling changes in user tasks and virtual machine availability. The Firefly Optimization uses 30 fireflies along with 100 generations and several attraction coefficients for optimizing cloudlet-based scheduling and Simulated Annealing enhances its search process via adjustable temperature (T) and cooling rates (α). The task scheduling approach was tested across dynamic and static scenarios with varying numbers of tasks (100–600) and virtual machines (VMs).The system performance is evaluated by measuring metrics like Makespan, Execution Time, Completion Time, Execution Cost, and Resource Utilization across different task and VM scenarios. Results indicate improved resource utilization and execution cost optimization for varying task and VM configurations.

Figure 1 flowchart depicts a mixed task assignment model that incorporates the firefly optimization technique alongside simulated annealing. It begins with the creation of fireflies (VMs) and input parameters, later involving the creation of other fireflies (VMs) and determination of execution cost (illumination). Neighbors are continuously recognized, attraction is determined, nodes are randomized to escape local optimum, execution costs are estimated, and VM-task mapping is optimized until the termination criterion is achieved.

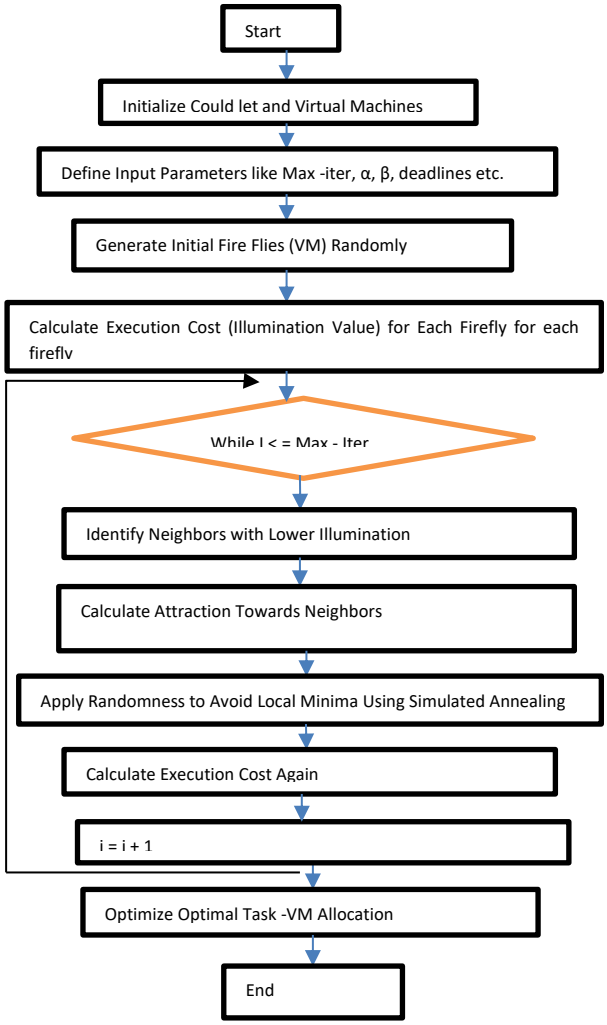

Fig.1. Proposed Methodology Flowchart

3.1 Implementation Algorithm

The suggested HOFO-SA method for task scheduling in cloud computing systems was covered in this section. The Hybrid Optimized Firefly Optimization with Simulated Annealing combines a strengths of the researchers improve optimization performance for complex cloudlet allocation through the combination of Firefly Optimization (FO) and Simulated Annealing (SA). The Firefly Optimization employs a population of fireflies (NUM_FIREFLIES = 30) that move towards brighter fireflies based on their attractiveness (BETA0 = 0.08) and the light absorption coefficient (GAMMA = 1.0). The algorithm functions across multiple dimensions (DIMENSIONS) by using parameters including displacement step (STEP_SIZE = 0.03) along with a number of neighbors (nt = 5) as well as decay rate (DECAY_RATE = 0.4) to find optimal solutions. The initial illumination value (lo = 0.05) helps in evaluating the attractiveness, and fireflies are attracted based on this illumination. Simultaneously, Simulated Annealing is incorporated to refine the optimization process by adjusting the temperature (T = 1) during each iteration, cooling down progressively (alpha = 0.9) until it reaches the final temperature (Tmin = 0.0001). The mechanism enables better exploration of global solutions compared to local minima through its probabilistic approach. The combination of these optimization algorithms enhances both the speed of convergence and precision thus becoming an effective solution for cloudlet allocation optimization issues and additional optimization problems. HOFO-SA improves cloud computing convergence times and stops systems from prematurely stalling while providing optimal scheduling of tasks. The method provides better load distribution along with shorter execution duration and more affordable operation in comparison to conventional approaches. Through its adaptive hybridization mechanism, the system increases exploration and exploitation capabilities, which results in high scalability together with reliable operation in dynamic environments.

The following proposed algorithms are used for this work:

A. Firefly Optimization (FO)

Cloud computing task scheduling using the Firefly optimization method is covered in this section. The suggested method was modeled using firefly optimization as it is appropriate for addressing the NP-hard issue, which is the precise equivalent of task scheduling in the cloud computing paradigm. It also takes fewer iterations, with fireflies traversing a large portion of the problem space. The method used in the firefly optimization technique is inspired by nature and simulates firefly behavior and attraction patterns using the torch patterns covered in [22]. The fundamentals of firefly optimization and how flashlights may draw two flies together are covered in [23]. Comparable to cloud computing's task scheduling issues, fireflies are better suited to solve NP-hard and multi-dimensional problems[24]. So, to address the issue of task scheduling in the cloud paradigm, this opted for the firefly optimization technique. It has been observed that the light of another fly may attract these fireflies. Their attraction will naturally be to the brightest flies.

In general, the following situations must be taken into account while using the firefly optimization approach: (1) using the assumption that all flies are of the same gender and will be attracted to one another regardless of gender. (2) The brighter firefly will attract a less bright one; in other words, appeal is exactly proportionate to brightness. (3) If a firefly is the brightest (best solution) among all fireflies, it moves randomly because there is no brighter firefly to attract it. This helps in exploring new areas in the search space. A firefly with higher brightness attracts others more strongly. Therefore, in order to go on with this strategy, it must determine the brightness, or the appeal and intensity [25].

The first step in determining a firefly's intensity is to use Equation (1):

$$Int(s) = \frac{Int(r)}{s^2} \quad (1)$$

The amount and quality of absorbed light determine an object's attractiveness. And so, there is a relationship between the light absorption coefficient and the brightness intensity. The result is shown in Equation (2).

$$Int = Int_0 . e^{-\Omega s^2} \quad (2)$$

It is necessary to determine the separation of two flies after determining their intensity. Equation (3) is used to compute this.

$$s = \left|x^i - x^j\right| = \frac{1}{r}\sqrt{\sum_{z=1}^{r}(x^{(i,z)} - x^{(j,z)})^2} \quad (3)$$

where r is the dimension of a firefly.

Following the distance calculation, this determined the population of fireflies, scattered them, and described the movement of a firefly i drawn to j, a firefly that is brighter than i, for the number of repetitions denoted by u. The firefly's movements are determined by Equation (4).

$$x_{u+1}^i = x_u^j + \Gamma_0 . e^{-\left\{\frac{s^2}{d^2}\right\}}\left(x_u^i - x_u^j\right) + \gamma EUR_i \quad (4)$$

Equation (5) may be used to get the randomized damping constant once the firefly's movement has been defined.

$$\gamma = \gamma_0 . \theta_u \quad (5)$$

where θ lies in between 0 and 1. θ is a damping constant.

B. Simulated Annealing

A metaheuristic optimization approach that has received a lot of attention is Simulated Annealing (SA), which was first presented in [26]. The SA algorithm outperforms conventional metaheuristic approaches by avoiding locally optimum solutions; it takes its cue from the annealing process, which is used by metals. This property helps SA stand out among other metaheuristic algorithms when it comes to solving optimization problems with complicated, non-convex solution spaces and less-than-ideal solutions[27]. Complex optimization issues may be solved using simulated annealing. This algorithm's big error margin is its key advantage over competing meta-heuristics. To rephrase, SA is well-known for its ability to discover the global solution to any NP-hard issue[28]. The SA method takes a random beginning value X and uses its surrounding area to generate a new solution Y. The next step in SA is used to compute the fitness value of X and Y; if Fit(Y) is less than or equal to Fit(X), then X is equal to Y. However, SA is able to substitute Y for X even if Y's fitness is not higher than X's. Equations 6 and 7 define the probability (p), which determines this[29]:

$$P = e^{-\Delta E/T} \quad (6)$$

$$\Delta E = \frac{Cost(X) - Cost\ (Y)}{Cost\ (Y)} \quad (7)$$

The temperature variable, denoted by T, should begin at a high value and gradually fall as the iterations continue. P represents the likelihood of using a different approach. Δ E is the name given to the disparity between the objective values of solution X and the proposed solution Y.

3.2 *Proposed Algorithm:* Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA)

Step1: Initialization

1. Initialize Cloud Environment – Set up cloudlets and virtual machines.
2. Define Input Parameters:
   a) Number of fireflies (NUM_FIREFLIES = 30)
   b) Dimensionality (DIMENSIONS = Number of cloudlets)
   c) Maximum iterations (MAX_GENERATIONS = 100)
3. Firefly movement parameters:
   - Attraction coefficient (BETA0 = 0.08)
   - Light absorption coefficient (GAMMA = 1.0)
   - Decay rate (DECAY_RATE = 0.4)
   - Displacement step size (STEP_SIZE = 0.03)
   - Number of neighbors (nt = 5)
   - Initial illumination value (lo = 0.05)
4. Simulated Annealing Parameters:
   - Initial temperature (T = 1)
   - Minimum temperature (Tmin = 0.0001)
   - Cooling rate (alpha = 0.9)
   - Boltzmann constant (Boltzmann Constant = 1.0)

Step 2: Initialize vm List (Virtual Machines) and ct List (Cloudlets)

Step 3: Compute Execution Time and Cost for VMs for each VM i:

1. Compute capacity[i] based on MIPS and PEs
2. Compute Execution Time (ET[i][j]) for each cloudlet
3. Compute Completion Time (CT[i][j])
4. Compute Execution Cost (EC[i][j])

Step 4: Firefly Initialization

1. Randomly generate an initial population of fireflies representing task allocations between lower bound and upper bound.
2. Compute the execution cost (illumination value) for each firefly.
3. Store initial luciferin value for each VM

Step 5: Iterative Optimization (While Loop)

4 Repeatwhileiteration count i ≤ MAX_GENERATIONS:

a) Update luciferin values for each VM

b) Identify neighbors with lower illumination values (better execution cost).

c) Compute attraction strength towards brighter fireflies.

d) Update the firefly positions (task allocation) based on attractiveness.

e) Apply Simulated Annealing:

- Introduce randomness to escape local minima.
- Accept solutions based on Boltzmann probability criterion: $P = e^{-\Delta E/(kT)}$
- Decrease temperature using: $T = T \times \alpha$

5 Recalculate the execution cost.

6 Perform load balancing correction for each VM

7 Increment iteration count i = i + 1.

Step 6: Final Optimization

1. After reaching the maximum iterations, select the optimal task-Virtual Machine allocation based on the best firefly position.

4. Results and Discussion

The suggested hybrid HOFO-SA algorithm for cloud computing task scheduling is shown here with experimental results. The method defines experimental conditions through setup, parameter values and efficiency assessment standards. A discussion of the proposed algorithm results through measures indicates the findings before performing a comparative examination with present models.

4.1 Experimental Setup

The experiments were conducted on a Dell PC that has a 1 TB hard drive, 16 GB of RAM, an Nvidia GeForce GTX 1650 GPU unit, and an Intel Core i5-10300H CPU (2.50 GHz). The developers used Java to execute the HOFO-SA algorithm because of its proven ability to process complex optimization issues effectively. The CloudSim framework simulated cloud data centers alongside the Eclipse IDE which served as the development environment for coding functions as well as debugging techniques and executable execution. The research used Windows 10 as its operating system to deliver stability during simulations and cloud infrastructure modeling, which relied on Java.

4.2 Performance Measures

The following performance measures are calculated for task scheduling in cloud computing[30].

1. Makespan

The maximum completion time is determined by Makespan by specifying the completing time of the previous task. The makespan is one of the most well-known optimization criteria when it comes to task scheduling. Equation (8) may be used to determine it:

$$Makespan = \max - \min_{taski}(Fn_{Time}) \quad (8)$$

Where, Fn_{Time}shows the finishing time of task i.

2. Execution Time (Total and Mean)

Total Execution Time is the sum of all individual task execution times, while the mean is their average. Equation (9) below may be used to compute it:

$$T_{exec} = \sum_{i=1}^{n} E_i \ and \ \bar{E} = \frac{1}{n}\sum_{i=1}^{n} E_i \ (Mean) \quad (9)$$

where E_i is the Execution Time of task i.

3. Completion Time (CT)

Total completion time sums all task finish times, and the mean is the average completion time. CT calculated as Equation (10):

$$T_{comp} = \sum_{i=1}^{n} C_i \quad (10)$$

4. Variance

Variance quantifies the spread of values around the mean, indicating dispersion. Variance is measured as follows in Equation (11):

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \quad (11)$$

where x_i are the observed values and μ is their mean Equation (12):-

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \quad (12)$$

5. Cost

"Cost" refers to the overall monetary outlay to the cloud service provider in relation to resource use. While customers of the cloud want to minimize costs via optimal utilization, cloud providers primarily seek to grow revenue and profit. The following Equation (13), which measures cost:

$$Cost = \sum_{resource\ i}(C_i * T_i) \quad (13)$$

Where, Ci denotes a cost of VMi per time unit and Ti represents the time for which VMi is utilized.

6. Average Resource Utilization

For cloud providers, one of the essential metrics is resource utilization or the rate at which their resources are used to generate revenue. The average utilization may be calculated using Equation(14):

$$Average\ Utilization = \sum_{i=1}^{n} \frac{Execution\ time\ of\ reosurce\ i}{Makesapn*n} \quad (14)$$

Where, n is the number of resources.

4.3 Results of proposed HOFO-SA Algorithm

This section provides the results of the proposed algorithm for task scheduling in cloud commuting. The hybrid approach is designed to dynamically adjust task allocation and optimize computational resources based on the workload characteristics. The methodology evaluates load balancing under three scenarios: Dynamic Load Balancing with Varying User Tasks & Fixed Virtual Machines, Dynamic Load Balancing with Fixed User Tasks & Varying Virtual Machines, and Static Load Balancing with Varying User Tasks & Fixed Virtual Machines. The number of tasks used for analysis includes 100 to 600, while the number of virtual machines (VMs) varies across 5, 10, 15, 20 and 25 configurations. Performance is measured using key parameters such as makespan, execution time, task completion time, variance, execution cost, and resource utilization. The hybrid HOFO-SA model dynamically optimizes scheduling, ensuring an equitable distribution of workloads across available cloud resources, thereby enhancing system efficiency and reducing computational overhead.

Table 2. Dynamic (With Varying User's Tasks & Fixed Virtual Machines) Load Balancing Using HOFO-SA

Number of Tasks	Maximum Makespan	Minimum Makespan	Mean Completion Time (μ)	Variance	Total Execution Cost	Resource Utilization
100	4.3	3.9	82.01472	5.1323	12.0022	95.4103
200	8.3	7.89	323.8170	10.243	47.9726	195.131
300	12.3	11.89	725.6671	15.455	107.950	294.991
400	16.29	15.88	1287.254	20.354	191.888	395.183
500	20.28	19.88	2008.160	25.413	299.724	495.050
600	24.29	23.88	2890.632	30.778	431.794	595.123

Table 2 presents the results of the HOFO-SA hybrid model for dynamic load balancing with varying user tasks and fixed virtual machines. As the number of tasks increases from 100 to 600, the maximum makespan rises from 4.3 to 24.29, while the minimum makespan increases from 3.9 to 23.88, ensuring efficient scheduling. The mean completion time (μ) grows from 82.01472 to 2890.632, reflecting the impact of higher workloads. Variance increases from 5.1323 to 30.778, indicating greater dispersion in task execution times. The total execution cost scales from 12.0022 to 431.794, proportional to workload expansion, while resource utilization improves significantly from 95.4103% to 595.123%, demonstrating the model's efficiency in distributing computational tasks across available cloud resources. These results validate the HOFO-SA model's capability to dynamically optimize scheduling, enhance system efficiency, and reduce computational overhead.

Table 3. Dynamic (With Fixed User's Tasks & Varying Virtual Machines) Load Balancing Using HOFO-SA

Number of VMs	Maximum Makespan	Minimum Makespan	Mean Completion Time (μ)	Variance	Total Execution Cost	Resource Utilization
5	4.3	3.9	82.01472	5.1323	12.0022	95.4103
10	2.3	1.9	21.02422	1.8351	6.00727	91.4653
15	2.1	1.1	9.80897	3.5473	4.11404	70.0841
20	1.3	0.9	5.51976	0.6659	3.01186	84.9717
25	1.1	0.7	3.6144	0.4837	2.41080	82.1753

Table 3 presents the results of the HOFO-SA hybrid model for dynamic load balancing with fixed user tasks and varying virtual machines (VMs). As the number of VMs increases from 5 to 25, the maximum makespan decreases significantly from 4.3 to 1.1, while the minimum makespan drops from 3.9 to 0.7, indicating improved efficiency with more computational resources. The mean completion time (μ) reduces from 82.01472 for 5 VMs to just 3.6144 for 25 VMs, highlighting faster task execution. Variance values fluctuate, starting at 5.1323 for 5 VMs, decreasing to 0.6659 at 20 VMs, and slightly increasing to 0.4837 at 25 VMs, ensuring stable task distribution. The total execution cost follows a declining trend, reducing from 12.0022 at 5 VMs to 2.4108 at 25 VMs, showcasing cost efficiency. Resource utilization remains high, starting at 95.4103% for 5 VMs, dropping to 70.0841% at 15 VMs due to increased idle capacity, and stabilizing around 82.1753% at 25 VMs. These results confirm that the HOFO-SA model effectively optimizes resource allocation, reducing execution time and cost while maintaining efficient workload distribution across VMs.

Table 4. Static (With Varying User's Tasks & Fixed Virtual Machines) Load Balancing Using HOFO-SA

Number of Tasks	Maximum Makespan	Minimum Makespan	Mean Completion Time (μ)	Variance	Total Execution Cost	Resource Utilization
100	4.3	3.9	82.01472	5.1323	12.0022	95.4103
200	8.3	7.89	323.817	10.243	47.9726	195.131
300	12.3	11.89	725.6671	15.455	107.950	294.991
400	16.29	15.88	1287.254	20.354	191.888	395.183
500	20.28	19.88	2008.160	25.413	299.724	495.050
600	24.29	23.88	2890.632	30.778	431.794	595.123

Table 4 presents the results of the HOFO-SA hybrid model for static load balancing, where the number of user tasks varies while the virtual machines remain fixed. As the number of tasks increases from 100 to 600, the maximum makespan rises from 4.3 to 24.29, while the minimum makespan follows a similar trend, increasing from 3.9 to 23.88, indicating higher processing time with more tasks. The mean completion time (μ) grows significantly from 82.01472 for 100 tasks to 2890.632 for 600 tasks, reflecting increased workload pressure on fixed computational resources. Variance values also increase from 5.1323 to 30.778, showing greater dispersion in task execution times. Similarly, the total execution cost scales from 12.0022 at 100 tasks to 431.794 at 600 tasks, demonstrating the computational expense of handling more tasks under static resource allocation. Resource utilization improves consistently, increasing from 95.4103% for 100 tasks to 595.123% for 600 tasks, indicating that the system efficiently maximizes its available resources despite the static VM configuration. These results highlight the effectiveness of the HOFO-SA model in maintaining workload distribution and performance even under static load-balancing conditions.

Table 5. Static (With Fixed User's Tasks & Varying Virtual Machines) Load Balancing Using HOFO-SA

Number of VMs	Maximum Makespan	Minimum Makespan	Mean Completion Time (μ)	Variance	Total Execution Cost	Resource Utilization
5	4.3	3.9	82.01472	5.1323	12.0022	95.4103
10	2.3	1.9	21.02422	1.8351	6.00727	91.4653
15	1.7	1.3	9.67536	2.2666	4.05391	85.4009
20	1.3	0.9	5.51976	0.6659	3.01186	84.9717
25	1.1	0.7	3.6144	0.4837	2.4108	82.1753

Table 5 presents the results of the HOFO-SA hybrid model for static load balancing with a fixed number of user tasks and varying virtual machines (VMs). As the number of VMs increases from 5 to 25, the maximum makespan decreases significantly from 4.3 to 1.1, while the minimum makespan reduces from 3.9 to 0.7, demonstrating improved efficiency as more computational resources become available. The mean completion time (μ) follows a downward trend, dropping from 82.01472 for 5 VMs to 3.6144 for 25 VMs, indicating faster task execution. Variance initially decreases from 5.1323 at 5 VMs to 0.6659 at 20 VMs, then slightly stabilizes at 0.4837 for 25 VMs, ensuring balanced workload distribution. The total execution cost also declines from 12.0022 at 5 VMs to 2.4108 at 25 VMs, emphasizing cost efficiency. The utilization pattern starts at 95.4103% with 5 VMs before stabilizing at 82.1753% with 25 VMs indicating efficient resource management. Static load balancing optimization results from implementing the HOFO-SA model due to confirmed effectiveness in scheduling tasks and allocating resources.

4.4 Comparison Between Existing and Proposed Algorithm

An extensive performance assessment demonstrates the distinctions between Firefly Swarm Optimization Task scheduling (FSOTS)[31]algorithm and its proposed Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA). The study shows enhancements in both makespan reduction and resource utilization efficiency results for different tasks running on various virtual machine setups.

Table 6. Comparison of average Makespan Between Existing and Proposed Algorithms on Different Tasks

Number of Tasks	FSOTS	Proposed HOFO-SA
100	4.43227	4.00073
200	8.33695	7.99542
300	12.32033	11.99445
400	17.69570	15.99068
500	21.63594	19.98160
600	25.03856	23.98860

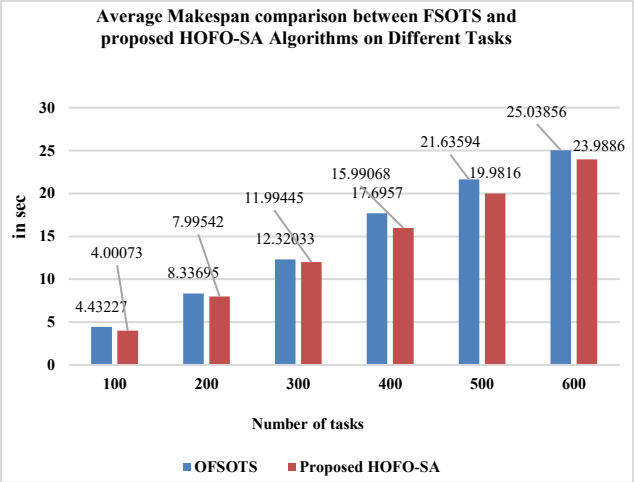

Fig.2. Bar graph of average Makespan comparison FSOTS and proposed HOFO-SA Algorithms on Different Tasks

This research demonstrates the difference in makespan performance between the existing FSOTS algorithm and the proposed Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA) approach for dynamic cloud computing load balancing through Table 6 and Figure 2. For 100 tasks, the existing algorithm has an average makespan of 4.43227, while the proposed algorithm performs slightly better with a makespan of 4.00073. At 200 tasks, the existing algorithm's makespan is 8.33695, compared to 7.99542 for the proposed algorithm. For 300 tasks, the existing algorithm takes 12.32033, and the proposed algorithm takes 11.99445. At 400 tasks, the existing algorithm's makespan increases to 17.69570, while the proposed one reaches 15.99068. With 500 tasks, the existing algorithm's makespan is 21.63594, and the proposed algorithm's makespan is 19.98160. Lastly, for 600 tasks, the

existing algorithm has a makespan of 25.03856, and the proposed algorithm shows an average makespan of 23.98860. These results show that the suggested approach reduces makespan more effectively than the current one, regardless of the size of the task.

Table 7. Comparison of Average Resource Utilization Ratio (ARUR) Ratio between Existing and Proposed Algorithms on Different Tasks

Number of Tasks	FSOTS	Proposed HOFO-SA
100	0.71920	0.95316
200	0.77149	0.97541
300	0.85816	0.98319
400	0.80871	0.98789
500	0.77598	0.99006
600	0.84505	0.99184

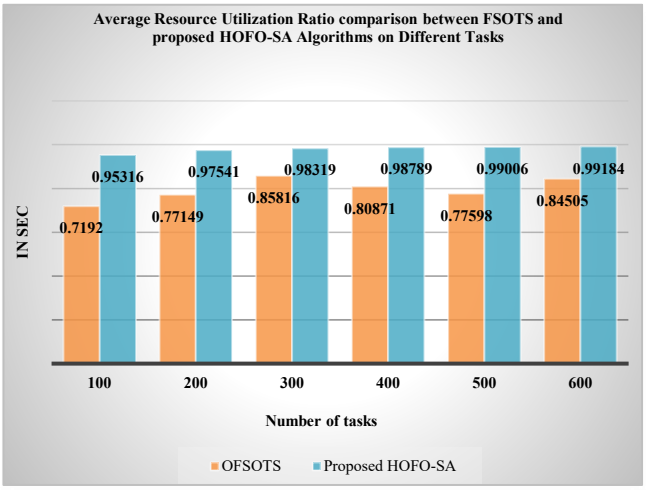

Fig.3. Bar graph of Resource Utilisation Ratio comparison FSOTS and proposed HOFO-SA Algorithms on Different Tasks

Table 7, along with Figure 3, demonstrates how the proposed HOFO-SA algorithm achieves better Average Resource Utilization Ratio (ARUR) outcomes than FSOTS for cloud computing load balancing while processing different task numbers. For 100 tasks, the existing algorithm achieves an RU of 0.71920, while the proposed algorithm performs significantly better with an RU of 0.953168. At 200 tasks, the existing algorithm's RU is 0.77149, compared to 0.97541 for the proposed algorithm. For 300 tasks, the existing algorithm achieves an ARUR of 0.85816, and the proposed algorithm reaches 0.98319. At 400 tasks, the existing algorithm's ARUR is 0.80871, while the proposed algorithm achieves 0.98789. With 500 tasks, the existing algorithm's ARUR is 0.77598, and the proposed algorithm's ARUR is 0.99006. Lastly, for 600 tasks, the existing algorithm achieves an ARUR of 0.84505, while the proposed algorithm reaches 0.99184. These results demonstrate that the proposed algorithm consistently outperforms the existing algorithm in optimizing resource utilization across all task sizes.

Table 8. Comparison of Average Makespan Between Existing and Proposed Algorithms for Different Virtual Machines (VMs)

Number of Tasks	FSOTS	Proposed HOFO-SA
5	4.26605	4.00073
10	2.14307	2.00242
15	1.49912	1.37134
20	1.23528	1.00395
25	1.06348	0.80359

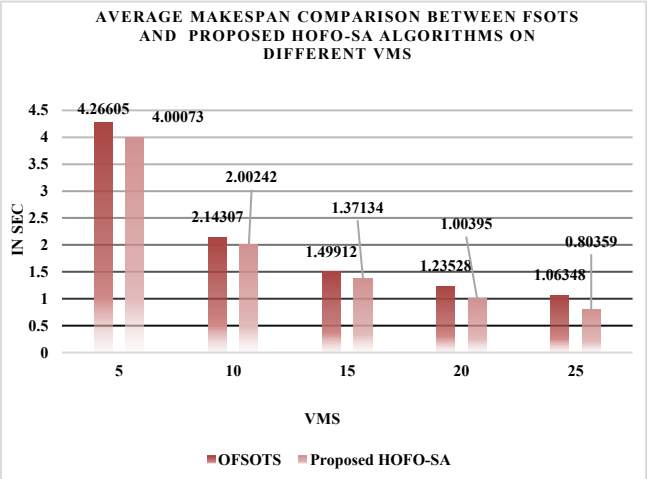

Fig.4. Comparative Analysis of Average Makespan between Existing and Proposed Algorithms on Different VMs

Table 8 and Figure 4 compare the Average Makespan between the existing algorithm FSOTS and the proposed algorithm (Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA)) for load balancing in cloud computing across varying numbers of VMs. For 5 VMs, the existing algorithm achieves an average makespan of 4.26605, while the proposed algorithm slightly improves with a makespan of 4.00073. For 10 VMs, the makespan for the existing algorithm is 2.14307, compared to 2.00242 for the proposed algorithm. At 15 VMs, the existing algorithm's makespan is 1.49912, while the proposed algorithm performs better with a makespan of 1.37134. With 20 VMs, the existing algorithm's makespan is 1.23528, and the

proposed algorithm achieves a lower makespan of 1.00395. Finally, for 25 VMs, the existing algorithm records a makespan of 1.06348, while the proposed algorithm outperforms it with a makespan of 0.80359. These results demonstrate how the suggested technique consistently outperforms all other evaluated VM setups when it comes to minimizing makespan.

Table 9. Comparison of Average Resource Utilization Ratio (ARUR) Between Existing and Proposed Algorithms on Different Virtual Machines (VMs)

Number of Tasks	FSOTS	Proposed HOFO-SA
5	0.80382	0.95316
10	0.63641	0.91291
15	0.62312	0.68242
20	0.47846	0.84664
25	0.42666	0.81811

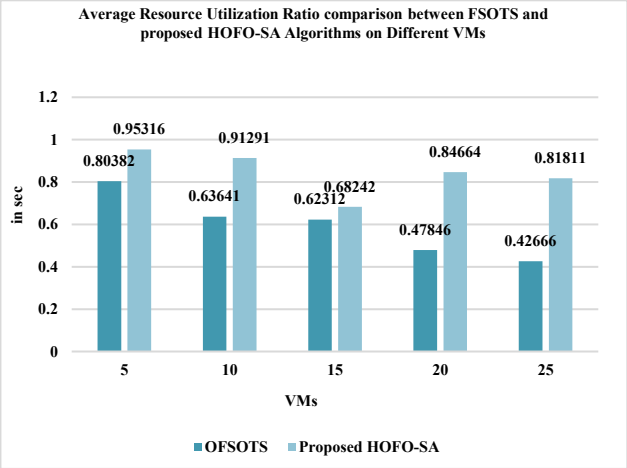

Fig.5. Comparative Analysis of Average Resource Utilization Ratio between Existing and Proposed Algorithms on Different VMs

Table 9 and Figure 5 compare the ARUR between the existing algorithm FSOTS and the proposed algorithm (HOFO-SA) for load balancing in cloud computing across varying numbers of VMs. For 5 VMs, the existing algorithm achieves an ARUR of 0.80382, while the proposed algorithm significantly improves it to 0.95316. For 10 VMs, the existing algorithm records an ARUR of 0.63641, compared to 0.91291 for the proposed algorithm. At 15 VMs, the ARUR for the existing algorithm is 0.62312, while the proposed algorithm achieves a slightly higher ratio of 0.68242. For 20 VMs, the ARUR of the existing algorithm decreases to 0.47846, whereas the proposed algorithm maintains a much higher ARUR of 0.84664. Lastly, for 25 VMs, the existing algorithm achieves an ARUR of 0.42666, while the proposed algorithm outperforms it with an ARUR of 0.81811. Across all evaluated virtual machine configurations, these findings show that the suggested technique is more efficient in using resources.

4.5 Discussion

HOFO-SA effectively schedules cloud computing tasks through the optimized firefly optimization search method, which maintains a balance of exploration and exploitation and through simulated annealing local search capabilities. The implementation of SA improves convergence speed and stops premature stagnation because it addresses standard heuristic failures. HOFO-SA beats standard FSOTS by delivering superior outcomes for minimal makespan together with reduced execution cost energy usage and efficient resource usage. The algorithm offers adaptive capabilities that match it perfectly to dynamic cloud environments and implements a dependable and scalable solution for real-time scheduling while achieving better load distribution alongside shorter execution times.

8 Conclusion And Future Work

The research presented the Hybrid Optimized Firefly Optimization with Simulated Annealing (HOFO-SA) algorithm to optimize cloud computing task scheduling.By combining Firefly Optimization (OFO) and Simulated Annealing (SA), the approach enhanced exploration and exploitation, leading to better task allocation and resource utilization. Experimental data shows that HOFO-SA algorithm provides better performance than the current FSOTS algorithm does. Resource utilization efficiency and makespan reduction serve as key outcomes from the comparative assessment. HOFO-SA achieved lower average makespan values than Optimized Firefly Optimization in all task and VM configurations creating a significant decrease from 25.03 to 23.98 for 600 tasks. The resource utilization ratio improved from 0.84 in FSOTS to 0.99 in HOFO-SA. The proposed hybridization method demonstrates efficient cloud environment adaptability, which leads to optimized scheduling and cost reduction benefits. The findings demonstrate that HOFO-SA achieves superior performance in managing extensive cloudlet allocation operations with effective scalability capabilities. Although it demonstrates effectiveness in practice the HOFO-SA algorithm also has certain operational limitations. The algorithm bases its analysis on an environment that remains unchanged which might inadequately address real-life workload fluctuations. Increasing the complexity of task sets creates a negative impact on real-time performance because of the way the algorithm operates. Future work will focus on integrating reinforcement learning and deep learning techniques to enhance adaptability in dynamic cloud environments.

Moreover, hybridizing HOFO-SfA with meta heuristic approaches like Genetic Algorithms or Particle Swarm Optimization could further optimize task scheduling. Implementing energy-aware scheduling and extending the model to edge and fog computing environments will also be explored to improve efficiency and scalability.

Author Contributions: Stefy Mathew contributed to the conceptualization of the research framework, development of the hybrid Firefly Optimization with Simulated Annealing model, and execution of experiments. She was also responsible for drafting the initial manuscript and integrating the technical components. Mahaveer Kumar Sain supervised the research, provided guidance on optimization strategies and cloud scheduling methodologies, and contributed to the analysis, interpretation, and refinement of the results. Both authors reviewed and approved the final version of the manuscript.

Data availability: Data available upon request.

Conflict of Interest: There is no conflict of Interest.

Funding: The research received no external funding.

Similarity checked: Yes.

References:

[1] M. B. Gawali and S. K. Shinde, "Task scheduling and resource allocation in cloud computing using a heuristic approach," *J. Cloud Comput.*, 2018, doi: 10.1186/s13677-018-0105-8.

[2] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Comput.*, 2022, doi: 10.1007/s00500-021-06488-5.

[3] A. Mantri, "An EHO-Grounded Task Planning to Improve Resource Application in Cloud Computing," *Int. J. Intell. Syst. Appl. Eng.*, 2023.

[4] S. Alsubai, H. Garg, and A. Alqahtani, "A Novel Hybrid MSA-CSA Algorithm for Cloud Computing Task Scheduling Problems," *Symmetry (Basel).*, 2023, doi: 10.3390/sym15101931.

[5] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Comput.*, 2023, doi: 10.1007/s10586-020-03221-z.

[6] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *J. Ambient Intell. Humaniz. Comput.*, 2023, doi: 10.1007/s12652-023-04541-9.

[7] N. Rana, M. S. Abd Latiff, S. M. Abdulhamid, and S. Misra, "A hybrid whale optimization algorithm with differential evolution optimization for multi-objective virtual machine scheduling in cloud computing," *Eng. Optim.*, 2022, doi: 10.1080/0305215X.2021.1969560.

[8] X. L. He, Y. Song, and R. V. Binsack, "The intelligent task scheduling algorithm in cloud computing with multistage optimization," *Int. J. Grid Distrib. Comput.*, 2016, doi: 10.14257/ijgdc.2016.9.4.28.

[9] J. Kakkottakath Valappil Thekkepuryil, D. P. Suseelan, Keerikkattil, and P. Mathew, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Comput.*, 2021, doi: 10.1007/s10586-021-03269-5.

[10] R. Masadeh, N. Alsharman, A. Sharieh, B. A. Mahafzah, and A. Abdulrahman, "Task scheduling on cloud computing based on sea lion optimization algorithm," *Int. J. Web Inf. Syst.*, 2021, doi: 10.1108/IJWIS-11-2020-0071.

[11] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *J. Ambient Intell. Humaniz. Comput.*, 2020, doi: 10.1007/s12652-020-02614-7.

[12] A. N. Malti, M. Hakem, and B. Benmammar, "A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems," *Cluster Comput.*, 2024, doi: 10.1007/s10586-023-04099-3.

[13] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment," *Int. Arab J. Inf. Technol.*, 2020, doi: 10.34028/iajit/17/1/9.

[14] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling," *Cluster Comput.*, 2021, doi: 10.1007/s10586-020-03205-z.

[15] P. S. Priya and S. J. J. Thangaraj, "Reducing Task Scheduling Time in Cloud Computing using Novel Improved Whale Optimization Algorithm over Ant Colony Algorithm," in *2024 International Conference on Trends in Quantum Computing and Emerging Business Technologies*, IEEE, Mar. 2024, pp. 1–5. doi: 10.1109/TQCEBT59414.2024.10545043.

[16] T. Ganesan, M. Almusawi, K. Sudhakar, B. R. Sathishkumar, and K. S. Kumar, "Resource Allocation and Task Scheduling in Cloud Computing Using Improved Bat and Modified Social Group Optimization," in *2024 Second International Conference on Networks, Multimedia and Information Technology (NMITCON)*, IEEE, Aug. 2024, pp. 1–5. doi: 10.1109/NMITCON62075.2024.10699250.

[17] P. V. A. Priyadarshini, Z. Ajzan Balassem, A. Seetha, P. Nandhini, and A. V Shreyas, "Dipper Throated Optimization with Convolutional Neural Network Based Task Scheduling and Resource Allocation in the Cloud Computing," in *2024 International Conference on Integrated Intelligence and Communication Systems (ICIICS)*, IEEE, Nov. 2024, pp. 1–5. doi: 10.1109/ICIICS63763.2024.10860170.

[18] H. Zhang and R. Jia, "Application of Chaotic Cat Swarm Optimization in Cloud Computing Multi Objective Task Scheduling," *IEEE Access*, vol. 11, pp. 95443–95454, 2023, doi: 10.1109/ACCESS.2023.3311028.

[19] J. Liu, D. Tang, J. Li, and Y. Xiao, "Cloud Computing Task Scheduling Based on Multi-strategy Improved Harris Hawks Optimization," in *2023 4th International Symposium on Computer Engineering and Intelligent Communications, ISCEIC 2023*, 2023. doi: 10.1109/ISCEIC59030.2023.10271142.

[20] S. Tian, S. Ji, and T. Wang, "Cloud Computing Scheduling Strategy Based on Adaptive Weighted Particle Swarm Optimization Algorithm," in *2023 4th International Symposium on Computer Engineering and Intelligent Communications, ISCEIC 2023*, 2023. doi: 10.1109/ISCEIC59030.2023.10271232.

[21] S. Srichandan and R. K. Jena, "Multi-objective Task Scheduling in Cloud Data Center using Cat Swarm Optimization Framework," in *2023 Asia Conference on Power, Energy Engineering and Computer Technology (PEECT)*, 2023, pp. 73–78. doi: 10.1109/PEECT59566.2023.00020.

[22] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," in *Studies in Computational Intelligence*, 2010. doi: 10.1007/978-3-642-12538-6_6.

[23] X. S. Yang, "Firefly algorithms for multimodal optimization," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009. doi: 10.1007/978-3-642-04944-6_14.

[24] J. Wu, Y. G. Wang, K. Burrage, Y. C. Tian, B. Lawson, and Z. Ding, "An improved firefly algorithm for global continuous optimization problems," *Expert Syst. Appl.*, 2020, doi: 10.1016/j.eswa.2020.113340.

[25] S. Mangalampalli, G. R. Karri, and A. A. Elngar, "An Efficient Trust-Aware Task Scheduling Algorithm in Cloud Computing Using Firefly Optimization," *Sensors*, 2023, doi: 10.3390/s23031384.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science (80-.).*, 1983, doi: 10.1126/science.220.4598.671.

[27] P. Gonzalez-Ayala, A. Alejo-Reyes, E. Cuevas, and A. Mendoza, "A Modified Simulated Annealing (MSA) Algorithm to Solve the Supplier Selection and Order Quantity Allocation Problem with Non-Linear Freight Rates," *Axioms*, 2023, doi: 10.3390/axioms12050459.

[28] M. Hanine and E. H. Benlahmar, "A load-balancing approach using an improved simulated annealing algorithm," J. Inf. Process. Syst., 2020, doi: 10.3745/JIPS.01.0050.

[29] I. Attiya, L. Abualigah, S. Alshathri, D. Elsadek, and M. A. Elaziz, "Dynamic Jellyfish Search Algorithm Based on Simulated Annealing and Disruption Operators for Global Optimization with Applications to Cloud Task Scheduling," *Mathematics*, 2022, doi: 10.3390/math10111894.

[30] N. Almezeini and A. Hafez, "Task Scheduling in Cloud Computing using Lion Optimization Algorithm," *Int. J. Adv. Comput. Sci. Appl.*, 2017, doi: 10.14569/ijacsa.2017.081110.

[31] D. C. Rao, S. Sharma, S. K. Nayak, S. K. Srichandan, and A. Dash, "A Novel Modified and Optimized Meta-Heuristic Load-Balancing Technique for Cloud Computing System," *Int. J. Intell. Syst. Appl. Eng.*, vol. 11, no. 9s, pp. 598–611, 2023.