International Journal of Current Engineering and Technology
©2025 INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

Research Article

An Efficient Bio-Inspired Optimization Framework for Scalable Task Scheduling in Cloud Computing Environments

Gopikrishna Maddali*

Independent Researcher

Received 01 May 2025, Accepted 15 May 2025, Available online 16 May 2025, **Vol.15, No.3 (May/June 2025)**

Abstract

The need for sharing and using resources is growing at a fast pace, which poses several issues for cloud computing (CC) as the number of users increases. For this reason, job scheduling with load balancing across resources is a crucial area for improving performance. High energy usage and underutilised resources are two of the major obstacles to effective task scheduling. To address this, propose a bio-inspired optimization framework utilizing the Lyrebird Falcon Optimization (LFO) algorithm, which mimics lyrebird behavior through two key phases: escaping (exploration) and hiding (exploitation). This population-based metaheuristic dynamically updates task assignments to minimize makespan and energy usage while enhancing CPU and resource utilization. The algorithm was implemented in CloudSim and evaluated across various task loads (1000–5000 tasks). Experimental results demonstrate that LFO consistently achieves lower makespan (from 22.13s to 18.78s) and energy consumption (from 21.67 kW to 23.70 kW) compared to the traditional Fruit Fly Optimization Algorithm (FOA), highlighting its efficiency. The key advantages of this work include its ability to minimize energy consumption while optimizing resource utilization, scalability to large-scale cloud environments, and improved performance, making it a promising solution for sustainable and efficient task scheduling in cloud computing.

Keywords: *Cloud Computing, Load Balancing, Task Scheduling, Bio-Inspired Optimization, Lyrebird Optimization Algorithm (LOA), Energy Consumption.*

1. Introduction

Cloud computing has matured into a distinct facet of Internet-based technology. Globally distributed, high-performance data centres are the backbone of legitimate cloud service providers like Amazon, Oracle, and Microsoft [1] [2]. The principles of utility computing form the basis of the cloud computing paradigm. In this model, IaaS, PaaS, and SaaS are service delivery methods that employ pay-as-you-go pricing and service-level agreements. The network, processing, and storage infrastructure resources are all combined into a single data centre unit [3] [4]. Virtual machines may take use of its massive processing power, storage capacity, and bandwidth. Systematic resource management primarily focusses on the virtual machine work assignment [5] [6] [7]. Due to the ever-increasing need for resource sharing and consumption, cloud computing encounters a barrage of problems as the user base grows [8] [9]. Consequently, a significant obstacle to task scheduling is load balancing [10] [11] [12].

*Corresponding author's ORCID ID: 0000-000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.15.3.4

The goal of load balancing is to utilise numerous resources as efficiently as possible with as little reaction time as possible, all while preventing any one resource from being overwhelmed [13] [14]. This is how load has to be distributed between resources in cloud-based building design, such that all resources are always doing the same amount of work [15] [16]. Resource supply and job scheduling are the two primary considerations of load balancing. As a result, it ensures that resources are readily available when needed and offers efficiently used resources [17] [18].

Therefore, a system that efficiently uses its resources, distributes its load evenly, and provides high-quality service rapidly while using as little time, energy, and money as possible requires an optimum task scheduling mechanism [19] [20]. Typically, the best schedules are generated using static, dynamic, and nature-inspired heuristic methods [21]. Thus, bio-inspired algorithms have surpassed traditional and exact methods as the superior choice [22]. Traditional bio-inspired algorithms like Genetic Algorithms (GA) [23], PSO [24], ACO [25], and Fruit Fly Optimization Algorithm (FOA) [26] They are widely used for task scheduling in cloud computing. Premature convergence, scaling problems, and inefficiency in very dynamic settings are common problems with these algorithms. GA evolves solutions using selection, crossover, and mutation, while PSO adjusts particle positions based on past best solutions. ACO simulates ant behavior to find optimal paths, and FOA mimics the behavior of fruit flies in foraging for food, focusing on optimizing tasks based on their sensory information [27] [28] [29]. These algorithms may get stuck in local optima, leading to suboptimal solutions, and struggle with managing large-scale tasks in cloud computing systems. Furthermore, it may require significant computation time and resources, particularly in complex, ever-changing cloud environments, limiting their effectiveness in real-time task scheduling [30]. The Lyrebird Falcon Optimization algorithm enhances conventional bio-inspired techniques by implementing a better method of handling exploration against exploitation. The system reduces premature convergence along with improving scalability which makes it suitable for handling large-scale as well as dynamic cloud computing requirements. LFO operates quickly to adapting environments thus generating efficient and accurate task scheduling results.

A. Motivation and Significance

Due to increasing scale and complexity of such systems, efficient scheduling of tasks in order to fully utilise resources is critical and hence scheduling algorithms are becoming vital in cloud computing. Traditional decision-making methods fail to share workloads equally that causes inefficiencies in duration and both resource usage and power utilization. It has been shown that bio-inspired optimisation algorithms may handle optimisation issues in contexts that are dynamically changing in near-optimal time, leading to their rising popularity. The research examines the necessity of developing an adaptive scheduling framework for cloud systems which provides scalability together with energy efficiencies and operational cost reduction benefits.

B. Contribution of study

This research proposes a bio-inspired optimization framework for scalable task scheduling in CC environments. The key contributions of this study include:

- Establishment of an optimization model using bio-inspired algorithms with the aim of improving the efficiency in task scheduling.
- Incorporation of performance-conscious scheduling and other cloud computing measures like makespan, Energy Consumption, CPU utilization and Resource Utilization.
- Designing and implementing an adaptive algorithm based on the biological processes that can be optimized as per the load.
- Thorough testing of the suggested architecture with several task loads (1000-5000 tasks) to confirm its efficacy.
- Comparison of the proposed bio-inspired approach with the typical scheduling strategies and techniques concerning the achieved Makespan, energy consumption and the resources utilization.

C. Novelty and Justification

This study is significant because it proposed the Lyrebird Falcon Optimisation (LFO) approach that is derived from the adaptive behaviours of lyrebirds with a view to solving the scaling of jobs in cloud computing systems. This bio-inspired strategy aims at combining the exploration and exploitation phases for making the makespan and Energy Consumption minimal while optimizing the CPU and Resource Utilization. The rationale for this work is to show how better in energy consumption compared to other techniques like the Fruit Fly Optimization Algorithm (FOA). Based on LFO's heedless controlling power to handle large-scale task loads and its opportunity of sustainability and utilization of resources, LFO is a substantial advancement in task scheduling in CC.

D. Structure of paper

The outline of the paper is as follows: Literature studies on bio-inspired optimisation for cloud task scheduling are included in Section II. The suggested structure is described in Section III. The experimental findings are presented in Section IV. Findings and recommendations for further study constitute Section V.

2. Literature Review

This section summarises current studies on bio-inspired algorithms for cloud computing workload scheduling and balancing. Task scheduling is a hot topic right now, and academics have been focusing on bio-inspired algorithms as a potential solution. This is all because cloud computing is becoming more popular among consumers.

Krishna and Khasim Vali, (2025) simulate and evaluate process scheduling options, we used WorkflowSim, a simulation toolset, and put the suggested method through its paces. Using scientific procedures such as Montage, Cybershake, SIPHT, and LIGO, ADWEH is tested against state-of-the-art algorithms such as Deep Q-Network (DQN), Advantage ActorCritic2 (AC2), and PWSA3C. In comparison to DQN, A2C, and PWSA3C methods, ADWEH achieves a significantly shorter makespan (by as much as 31.9%), lowers energy consumption (by as much as 24%), improves scalability efficiency (by about 29%), increases resource utilisation (by about 52%), and reduces failure rates (by more than 47%). The findings establish ADWEH as a strong and effective tool for enhancing an efficiency and dependability of processes in the cloud, meeting the changing needs of contemporary cloud settings [31].

Naveen et al. (2025), introduced the Butterfly Ant Colony Optimization Algorithm (BACOA), a novel hybrid approach designed to enhance local and global search capabilities. BACOA combines the exploratory strength of the BOA and the exploitation capabilities of ACO to minimize energy consumption, communication cost, and computation cost in Task Scheduling (TS). By balancing these parameters, BACOA achieves a more energy-efficient task-scheduling process while reducing the overhead costs associated with communication and computation in cloud systems. The simulation study shows that BACOA is better than other existing algorithms by achieving better energy saving by 16%, while it provides better communication cost by 21.06% and better computation cost by 18%, and it is more appropriate for large-scale cloud environments [32].

Choppara and Sudheer (2025), included creating a framework for adaptive scheduling that can accommodate scheduling based on dependencies, as well as sequential and parallel processes. By using this framework, many important performance metrics are enhanced by 30%, makespan is decreased by 25%, fault tolerance is enhanced by 25%, and system scalability and reliability are enhanced by 20%. Our method for data processing localisation increases data privacy and management efficiency while reducing latency and bandwidth utilisation by as much as 40%. The findings back up the promise mix of sophisticated machine learning with fog computing, as Simpy recreated this work using datasets from Google Cloud Jobs [33].

K. Singh and Bharti (2024), applied an optimization mechanism to improve the efficiency of the model based on the problems that occur during the allocation of resources and balancing mechanisms. So, to analyze the complete scheduling mechanism and problem identification mechanism, the concept of Back-Propagation Algorithm (BPA) with ANN is used, which is named ANN-BPA. In the end, to validate the model efficiency, a comparative analysis is performed with several swarm algorithms like PSO, ABC, CSA and MFO. In this case, the energy consumption, job completion rate, execution time, and overall performance are all improved by combining MFO with ANN-BPA [34].

Marathe et al., (2024), introduced a novel approach using the Dingo Optimization Algorithm (DOA), a multi-objective meta-heuristic scheduling technique designed to address key challenges in cloud resource management. The DOA focuses on optimizing resource utilization by balancing load distribution, reducing energy consumption, and minimizing task completion time. Simulation results demonstrate that DOA outperforms traditional algorithms by achieving a 20% improvement in Resource Utilization, a 17% reduction in Energy Consumption, and a 15% decrease in task completion time [35].

Bennett et al. (2024), proposed a solution based on energy consumption and resource allocation that schedules the work in a manner that allowed for consideration of the amount of load and the environment. Computer simulations show that the HBA performs better than the PSO, GA and the IDOA with energy consumption cut by 15.6 %, resource usage optimized by 18.2%, time taken to complete the tasks improved by 17.4% and the overall throughput improved by 21.2% meaning that HBA is a promising solution for efficient and scalable CFCs [36].

Rajasingh and Durga (2024), proposed an Energy Efficient Maximal Support Priority Scheduling Approach (EEMSPS), which schedules tasks in a virtual cloud environment. Furthermore, the Task Completion Time (TCT) algorithm is utilized to calculate the task length and time. Additionally, the SBABC method is employed to organize the task and reduce data processing time. Finally, cloud load balancing is evaluated using the EEMSPS algorithm. Energy efficiency, task execution time, and energy consumption are used to compare the performance of the proposed methodology to those of traditional approaches. Thus, cloud job scheduling achieves an energy efficiency rating of 96.4% [37].

Ramesh et al., (2023), particle swarm optimisation (PSO) in a multi-cloud environment to map intricate processes to various cloud services, including computation and storage. They connect complicated processes to storage services and serverless platforms offered by well-known cloud providers, like AWS, Azure, and GCP. The experimental assessment reveals that our method outperforms cloud-based naïve and intuition-based mapping by as much as 61% in terms of makespan and 51% in terms of cost of process setup [38].

Table 1 Summary of the related work for task scheduling in cloud computing

Ref	Methodology	Findings	Advantages	Challenges / Research Gaps	Recommendations
Krishna & Khasim Vali (2025)	ADWEH with WorkflowSim simulation, compared with DQN, A2C, PWSA3C	Up to 31.9% reduced makespan, 24% less energy use, 52% resource utilization	Robust in performance, scalability, fault tolerance	Limited to simulation; real-cloud deployment missing	Test ADWEH on hybrid/multi-cloud real-time datasets
Naveen et al. (2025)	Butterfly Ant Colony Optimization Algorithm (BACOA)	Energy savings (16%), 21.06% less comm. cost, 18% less computation cost	Strong global-local search; low energy/overhead	Complex hybridization may need adaptive tuning	Investigate BACOA with dynamic workloads and fog systems

	combining BOA & ACO				
Choppara & Sudheer (2025)	Adaptive scheduling using ML + Fog computing, SimPy on Google Cloud Jobs	30% perf. improvement, 25% less fault rate, 40% lower bandwidth usage	Supports all scheduling types; privacy-enhanced	Lack of support for multi-tenant or multi-cloud scenarios	Incorporate reinforcement learning for predictive scheduling
K. Singh & Bharti (2024)	ANN with Back Propagation (ANN-BPA) + Swarm Optimization (MFO, etc.)	Better than PSO, ABC, CSA, MFO in exec. time, energy use	Strong hybrid AI model; dynamic feature selection	Model depends on proper feature tuning	Extend BPA-ANN for real-time auto-scaling workloads
Marathe et al. (2024)	Dingo Optimization Algorithm (DOA), multi-objective metaheuristic	20% higher utilization, 17% energy saved, 15% faster	Dynamic load balancing, scalable TS	Limited comparison with other modern deep RL methods	Combine DOA with deep learning for adaptive learning
Bennett et al. (2024)	HBA - task prioritization with energy/resource demand adaptation	Energy down 15.6%, 18.2% more resource use, throughput up 21.2%	Adaptive to workload, environment	Evaluation limited to homogeneous cloud settings	Apply to heterogeneous cloud + edge scenarios
Rajasingh & Durga (2024)	EEMSPS + SBABC + TCT algorithm	96.4% energy efficiency in scheduling	Effective task grouping, strong load balancing	Focused only on energy; lacks security/latency factors	Explore EEMSPS in security-constrained multi-clouds
Ramesh et al. (2023)	PSO for workflow mapping on AWS, GCP, Azure	61% improved makespan, 51% cost reduction	Multi-cloud compatibility, low deployment cost	Not tested for real-time latency-critical jobs	Extend PSO mapping for latency-aware or burst workloads

3. Methods and Materials

The objective of this project is to provide a framework for optimising cloud computing task scheduling that is bio-inspired and efficient. The Lyrebird Falcon Optimization (LFO) algorithm is a population-based metaheuristic approach designed for task scheduling in CC environments. It simulates the behavior of lyrebirds during danger situations, where the population members, representing candidate solutions, update their positions in the problem-solving space based on two phases: escaping (exploration) and hiding (exploitation). The position of each lyrebird is updated iteratively using random selection between these strategies, aiming to minimize makespan and energy consumption while optimizing CPU and resource utilization. The algorithm initializes a population matrix of lyrebirds, evaluates their objective functions, and iteratively updates their positions based on the defined strategies. The objective function that has been assessed most effectively is used to choose the optimal solution. We employ CloudSim to create a cloud environment simulation to evaluate the algorithm's 1000–5000 job performance based on makespan, Energy Consumption, CPU utilisation, and resource utilization. The results indicate that LFO effectively reduces makespan and energy consumption while managing resource utilization across different task intensities, demonstrating its potential in efficient cloud task scheduling. The following Figure 1 flowchart illustrates all the steps involved in the implementation of Task Scheduling in Cloud Computing.

Proposed LFO algorithm

Lyrebirds are the building blocks of the LOA approach, a population-based metaheuristic algorithm. Finding suitable solutions to optimisation difficulties inside an iteration-based approach may be possible with the use of the LOA's collective search capabilities in the problem-solving area. Each lyrebird in the LOA utilises its location in the solution space to ascertain the decision variables' values [39]. From a mathematical perspective, it is possible to describe each lyrebird as a vector, with each member representing a decision variable. A theoretical matrix representation of the algorithm's population, consisting of LOA members, may be obtained by using Equation (1). The members of the LOA are first placed in the problem-solving area at random using Equation (2).

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_i \\ \vdots \\ X_N \end{bmatrix}_{N\times m} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} & \cdots & x_{1,m} \\ \vdots & \ddots & \vdots & ⋰ & \vdots \\ x_{i,1} & \cdots & x_{i,d} & \cdots & x_{i,m} \\ \vdots & \ddots & \vdots & ⋰ & \vdots \\ x_{N,1} & \cdots & x_{N,d} & \cdots & x_{N,m} \end{bmatrix}_{N\times m} \quad (1)$$

$$x_{i,d} = lb_d + r.(ub_d - lb_d) \quad (2)$$

Here, X is the LOA population matrix, X_i is the ith LOA member (Candidate Solution), $x_{i,d}$ is the decision variable's dth dimension in the search space, N is the lyrebird count, m is the decision variable count, and r is an interval-based random integer[0,1], the lower limit of the dth decision variable is denoted as lb_d and the upper bound as ub_d.

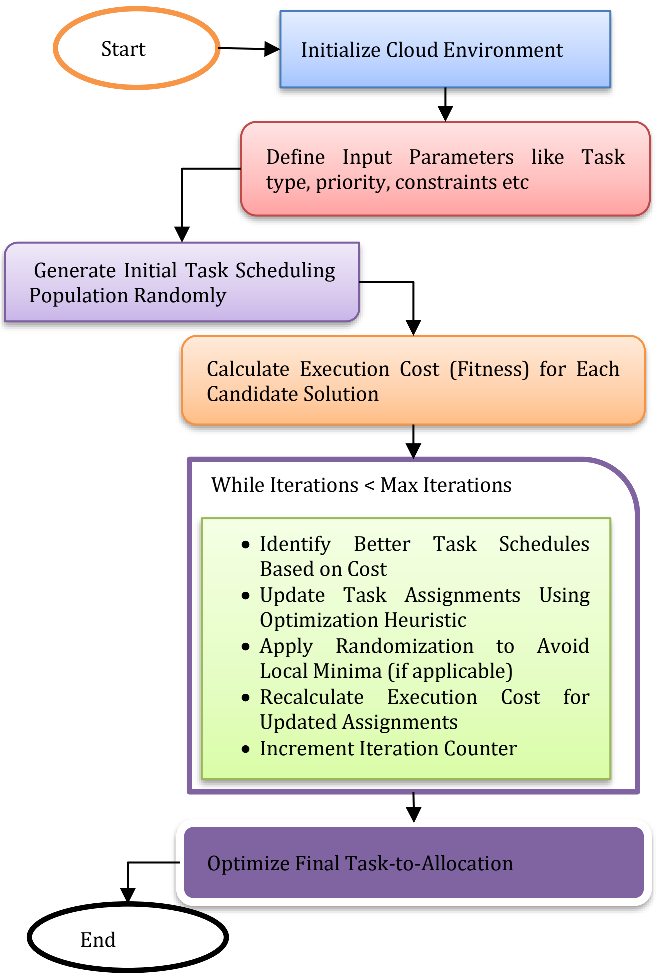
Start
Initialize Cloud Environment
Define Input Parameters like Task type, priority, constraints etc
Generate Initial Task Scheduling Population Randomly
Calculate Execution Cost (Fitness) for Each Candidate Solution
While Iterations < Max Iterations
Identify Better Task Schedules Based on Cost
Update Task Assignments Using Optimization Heuristic
Apply Randomization to Avoid Local Minima (if applicable)
Recalculate Execution Cost for Updated Assignments
Increment Iteration Counter
Optimize Final Task-to-Allocation
End

Figure 1 Proposed methodology flowchart for task scheduling in cloud computing

In order to assess the problem's objective function, it is helpful to remember that each LOA member stands for a possible remedy. Consequently, the values for the goal function are accessible in proportion to the population size. The set of values that have been assessed for the objective function of the problem may be shown as a vector using Equation (3).

$$F = \begin{bmatrix} F_1 \\ \vdots \\ F_i \\ \vdots \\ F_N \end{bmatrix}_{N\times m} = \begin{bmatrix} F(X_1) \\ \vdots \\ F(X_i) \\ \vdots \\ F(X_N) \end{bmatrix}_{N\times m} \quad (3)$$

In this context, F represents the evaluated objective function vector and F_i Stands for the evaluated objective function according to the ith member of the LOA.

An appropriate metric for assessing the solutions that have been suggested is the assessed values of the objective function. Therefore, the optimal candidate solution (or LOA member) is represented by the objective function's best evaluated value, whereas the optimal candidate solution (or LOA member) is represented by the objective function's worst evaluated value. Considering that the lyrebirds' location in the problem-solving space changes with every iteration, it is sensible to compare the objective function values and update the best candidate solution.

Mathematical Modelling of LOA

The suggested LOA method incorporates mathematical modelling of the lyrebird technique for threat detection into its design, which updates the population's location in each iteration. Depending on the lyrebird's decision in this case, the two processes of updating the population are (i) fleeing and (ii) hiding. To simulate the lyrebird's ability to decide between a hiding and an escape plan in the event of danger, LOA incorporates Equation (4) into its design. In other words, each iteration only makes use of one of the two phases to update the position of every LOA member.

$$Update\ process\ for\ X_i\colon \begin{cases} based\ on\ Phase\ 1, r_p \le 0.5 \\ based\ on\ Phase\ 2, else \end{cases} \quad (4)$$

Here, r_p is an integer chosen at random from the set [0, 1].

Phase 1: Escaping Strategy (Exploration Phase)

In particular, when the virtual lyrebird leaves the dangerous area for the safe ones, LOA changes the population member's position in the search space accordingly. An example of LOA's exploration skills in global search may be seen in the lyrebird's ability to alter its position and investigate new areas of the problem-solving region after being moved to a safe place. A member's safe zone in LOA design is the region around other members of the population with higher objective function values. By using Equation (5), we can therefore ascertain the collection of protected regions for every member of the LOA.

$$SA_i = \{X_k, F_k < F_i\ and\ K \in \{1,2,..,N\}\},\ \ where\ i = 1,2,..,N, \quad (5)$$

Here, SA_i is the group of protected spots for the ith lyrebird and X_k Represents the kth row of the a matrix, which outperforms the ith LOA member in terms of the objective function value (F_k) (i.e., X_k<F_i).

It is believed that the lyrebird would haphazardly make its way to one of these protected areas according to the LOA design. At this stage, we utilise the lyrebird displacement model to formulate Equation(6), which we then apply to find the new locations of all LOA members. Next, this new location will take the place of the old one of the matching member in accordance with Equation (7) if the goal function's value is enhanced.

$$x_{i,j}^{P1} = x_{i,j} + r_{i,j}.(SSA_{i,j} - l_{i,j}.x_{i,j})\ (6)$$

$$X_i = \begin{cases} X_i^{P1}, F_i^{P1} \le F_i \\ X_i, else \end{cases}, \quad (7)$$

Here, SSA_i is the chosen secure location for the ith lyrebird $SSA_{i,j}$ is its jth dimension, X_i^{P1} computes the new location of the ith lyrebird using the escape plan for the intended LOA, $x_{i,j}^{P1}$ is its $j$thdimension, F_i^{P1} is the value of its objective function, $r_{i,j}$ are arbitrary integers drawn at random from the range [0, 1] and $l_{i,j}$ are integers chosen at random to be either 1 or 2.

Phase 2: Hiding Strategy (Exploitation Phase)

At this point in LOA, the lyrebird's model for concealing in its immediate safe zone informs the process of updating the population member's location in the search landscape. Lyrebirds are able to use LOA in local searches by precisely assessing their environments and taking little steps to seek an appropriate hiding place, which causes them to make modest movements in their location.

Using the lyrebird's predicted path to a nearby hiding spot as input, LOA designers may update the positions of all LOA members according to Equation (8). The related member's previous position will be replaced if this new placement raises the value of the objective function according to Equation (9).

$$x_{i,j}^{P2} = x_{i,j} + (1 - 2r_{i,j}).\frac{ub_j - lb_j}{t} \qquad (8)$$

$$X_i = \begin{cases} X_i^{P2}, F_i^{P2} \le F_i \\ \quad X_i, else \end{cases} \qquad (9)$$

Here, X_i^{P2} is the new spot for the ith lyrebird found by using the concealed approach of the planned LOA, $x_{i,j}^{P2}$ is its jth dimension, F_i^{P2} is its objective function value, $r_{i,j}$ are arbitrary integers drawn at random from the range [0, 1] and t is the IterationCounter.

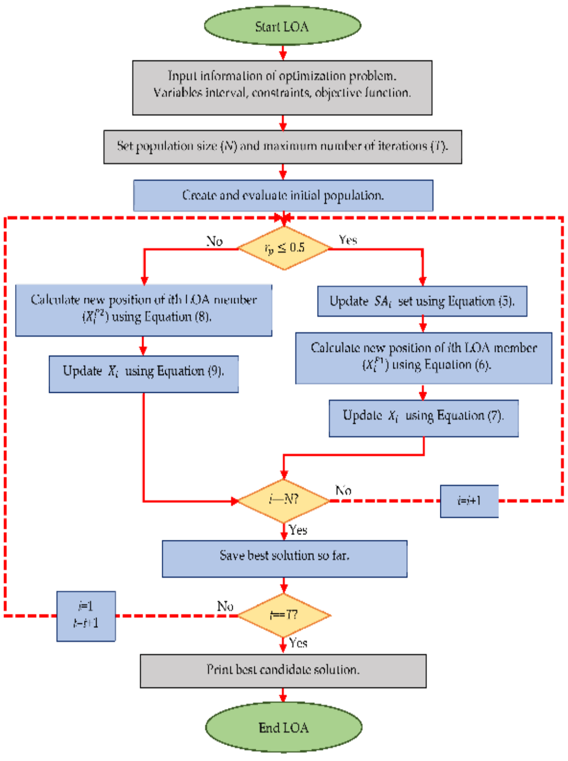
Start LOA
Input information of optimization problem. Variables interval, constraints, objective function.
Set population size (N) and maximum number of iterations (T).
Create and evaluate initial population.
No
$r_p \le 0.5$
Yes
Calculate new position of ith LOA member (X_i^{P2}) using Equation (8).
Update X_i using Equation (9).
Update SA_i set using Equation (5).
Calculate new position of ith LOA member (X_i^{P1}) using Equation (6).
Update X_i using Equation (7).
i==N?
No
i=i+1
Yes
Save best solution so far.
i=1
t=t+1
No
t==T?
Yes
Print best candidate solution.
End LOA

Figure 2 Flowchart of LOA.

Figure 2 shows a flowchart of the LOA implementation processes. Following the steps outlined in the LOA flow diagram is the procedure: An algorithm is first provided task-specific data, including the objective function, restrictions, and decision variables. Counting the number of iterations and members of the population is the next stage in solving the problem. In the first step, the objective function of the problem is used to evaluate a randomly selected beginning population for the algorithm. Following the initialisation step, the algorithm begins its first iteration. Next, move the original lyrebird to a different spot in the problem-solving area. According to LOA modelling, the lyrebird uses two tactics when it is in danger: (i) run away and (ii) hide. Equation (4) states that the LOA design assumes that each lyrebird has an equal probability of selecting one of these two strategies at random. If the lyrebird decides to choose the escape option, it may move about in the problem-solving area using Equations (5)–(7). In order for the lyrebird's location in the problem-solving space to be updated, it is necessary to follow equations (8) and (9). At this point in time, we have effectively adjusted the population's location for the first lyrebird. Adding the whereabouts of the other lyrebirds to the problem-solving area is done in the same manner as the first lyrebird. After repositioning every lyrebird in the solution space, the first iteration of the algorithm is finished. Iteratively comparing evaluated values for the target function allows us to establish the best potential solution up to this point. The program then employs the same strategy for each iteration until the last iteration, updating lyrebirds in the problem-solving space repeatedly. The output will include the solution to the issue that was achieved throughout the iterations of the algorithm after all of the iterations have been completed. Here, the algorithm's implementation is considered successful.

Performance Metrics

Task scheduling in CC is calculated using the following performance metrics.

Makespan

The time needed to do each action or operation in a system is called its makespan. When planning and managing jobs with the end objective of completing them as quickly as possible, it is crucial. Equation 10 allows for its calculation:

$$Makespan = \max - \min_{taski}(Fn_{Time}) \qquad (10)$$

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

Energy Consumption

The reduction of cloud infrastructure's overall power usage is the primary objective of EnergyConsumption. To be sustainable and cost-effective, it is essential.

Some strategies for optimising energy usage include distributing workloads, consolidating servers, and using dynamic resource scaling. It may be determined by plugging the values into equation 11:

$$Energy\ Consumption\ (E) = \sum_{t-1}^{n} P_i \times T_i \qquad (11)$$

Where: P_i = Power consumed by server i (in Watts), T_i = Time duration server i is active (in hours) and n = Number of servers.

CPU Utilization

The efficiency of CPU usage in cloud computing setups may be measured by looking at CPU utilisation. We can prevent some PMs or VMs from being overworked while others are idle by balancing CPU utilization. Implementing efficient CPU utilisation practices may help prevent performance bottlenecks and optimise the use of hardware resources. As an example, consider the following equation (12):

$$CPU\ Utilization\ (\%) = \left(\frac{CPU\ Time\ Used}{Total\ Available\ CPU\ Time}\right) \times 100 \qquad (12)$$

Resource Utilization

Measures how efficiently computational resources are used. It is computed as the ratio of utilized resources to the total available capacity. Higher resource utilization ensures optimal workload distribution across resources. Balancing resource usage prevents bottlenecks and enhances system efficiency. The expression for resource utilization, Ru, is Eq. (8):

$$R_u = \frac{T_c}{M_a\ x\ N} \qquad (13)$$

where T_c reflects the time required to execute an activity, M_a represents makespan and N represents number of resources.

4. Result analysis and Discussion

An Intel(R) Core (TM) i5-3230 M, 2.60 GHz, 8 GB of RAM, 750 GB of hard drive space, Windows 10, NetBeansIDE, JDK 8.0, and CloudSim make up the experimental environment used in this study. The CloudSim toolkit was used to implement the energy-aware techniques, dynamic scheduling system, and mechanism for recognising and grouping similar kinds of workloads.

Table 2 details the experimental setup

Experimental setup

Experimental Setup	Description
SimulationToolkit	CloudSim 5.0
CloudEnvironmentType	Simulated
PMs	1
Number of Tasks	1000 to 5000

Table II provides details of the experimental setup used for evaluating the LFO (Lyrebird Falcon Optimization) approach. The experiments were conducted with CloudSim 5.0 serving as the simulation toolkit to model the CC environment. The cloud environment type is simulated, ensuring a controlled testing scenario. The setup includes a single physical machine (PM), and the number of tasks varies from 1000 to 5000 to analyze performance across different workload intensities.

Table 2 LFO performance for various task counts

Task Count	Makespan (in sec)	EC (kW)	CPU-U (%)	R U (in%)
1000	22.1382	21.6782	24.9813	40.7014
2000	19.0584	20.5181	31.0325	40.5618
3000	18.7871	22.6694	14.4126	29.6925
4000	18.7893	25.1941	17.7209	27.7893
5000	18.7883	23.7029	21.7924	23.8816

Table III demonstrates how the Lyrebird Falcon Optimization (LFO) algorithm performs when scheduling tasks while running in a CC setting as the task numbers fluctuate. The makespan diminishes steadily from 1000 to 5000 tasks while attaining a stable value of 18.78 seconds with 3000 to 5000 tasks because of effective parallel scheduling. The studied algorithm demonstrates consistent energy consumption during execution that slightly changes yet reaches 25.19 kW at 4000 tasks while managing energy distribution across workloads. The CPU usage (CPU-U) maintained changes through 31.03% peak utilization at 2000 tasks then decreased to 14.41% at 3000 tasks before it rose gradually. The variable nature of measurements indicates that the algorithm adjusts its distribution process across virtual machines dynamically. The resource utilization (RU) decreases from 40.7% at 1000 tasks to 23.88% at 5000 tasks because of improved distribution of tasks and optimized resource allocation. The research findings show that LFO functions to preserve short makespan along with reasonable energy usage while it adjusts CPU and resource allocation according to workload strength.

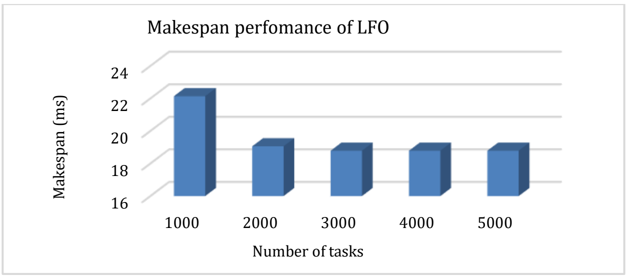

Figure 3 Flowchart of LOA Task Count with Makespan of LFO

The LFO algorithm demonstrates efficient and scalable performance through its steady makespan values that appear in Figure 3. As stated and illustrated in the previous sections, it can effectively arrange job schedules in a cloud computing environment to save many resources and time to complete tasks.

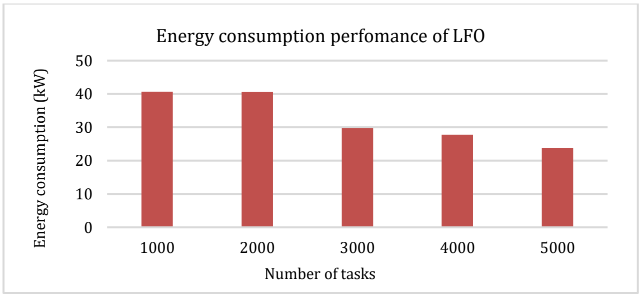

Figure 4 Task Count with Energy Consumption of LFO

Figure 4 shows distribution of the energy consumption or the LFO algorithm when the number of tasks increases and thus shows how to effectively control use of energy in cloud computing. Therefore, the workload distribution and input/output requirements are equally manageable as opposed to in other algorithms, which may eat up a lot of energy while handling a larger workload.

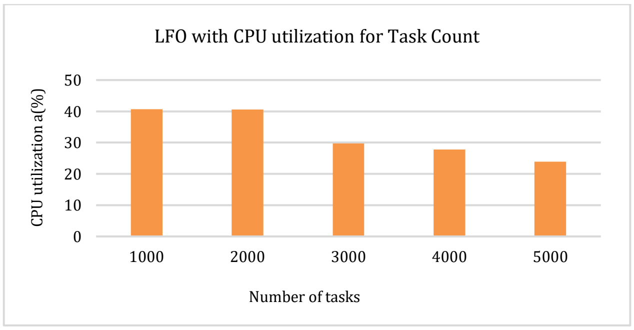

Figure 5 Task count with CPU Utilization of LFO

Figure 5 depicts the CPU usage of the LFO algorithm when the number of tasks rising and shows its effectiveness in managing the load among virtual machines. This makes sure that there is no resource that is overloaded or underutilized, making each one run at its best. This approach has the benefit of avoiding bottlenecks, utilizing the fastest method of data processing, and getting the optimal utilization out of current CPU cores especially important for cloud computing model of the systems.

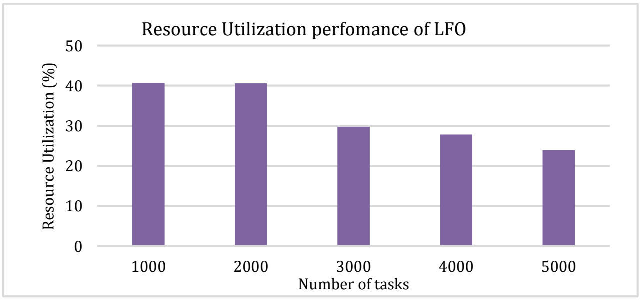

Figure 6 Task Count with Resource Utilization of LFO

Figure 6 depicts the fact that LFO algorithm effectively handles resource usage to accomplish a large number of tasks incrementally. It learns about the amount of work requested and performs the necessary allocations and deallocations effectively controlling for resource wastage. Organizational performance is improved, cost is reduced, and the use of computers is optimized when it comes to overall system operation. In the case of cloud computing, LFO is best suited for dynamically managing system resources, and this ensures sustainable and balanced job scheduling of all computational processes.

Comparison And Discussion

The suggested LFO algorithm is clearly more efficient than the previously utilised FOA approach for energy usage in cloud computing, as shown by the comparison. Regardless of the load on the tasks, LFO consumes less energy demonstrating a more efficient usage of computational necessities and electric power. This increase is due to LFO's ability to schedule tasks well and dynamically allocate its resources in a way that does not waste energy and thus is sustainable. The results validate that LFO offers a significant advancement over FOA in minimizing energy consumption in cloud environments.

Table 4 Comparison between previous work and our work for Performance Metrics of Cloud Computing

Energy Consumption		
Task	**LFO**	**FOA[40]**
1000	21.6782	79.8524
2000	20.5181	80.3654
3000	22.6694	82.9542
4000	25.1941	84.7429
5000	23.7029	86.9845

Table IV shows an energy consumption comparison between LFO strategies and FOA procedures when executing different task numbers. Task number increases from 1000 to 5000 with energy consumption percentage given on the y-axis while the x-axis shows the task count in Table IV. Energy use during the LFO process shows consistently better results than FOA method uses. For instance, at 1000 tasks, LFO consumes only 21.6782%, whereas FOA consumes 79.8524%. The energy costs remain stable at 23.7029% when LFO completes 5000 tasks yet FOA requires 86.9845% energy consumption in the same scenario. The research shows LFO uses significantly less energy than FOA at every task count which demonstrates its excellence for maximizing cloud resource management efficiency.

Key performance issues such excessive energy consumption, poor resource utilisation, and extended makespan are addressed in the proposed study by introducing the Lyrebird Falcon Optimisation (LFO) algorithm for task scheduling in CC. The simulation of lyrebird strategies within LFO helps achieve workload balance for enhanced system performance. This performance is compared with other existing methods for the efficiency that has been observed for energy consumption, namely CPU utilization and overall resources, for which it can be considered as nearly optimal, making it a sustainable and scalable technique to apply to current cloud services.

Conclusion and Future Scope

Issues with resource sharing and utilisation are becoming more problematic for cloud computing due to the increasing number of users. Consequently, a crucial area for scheduling jobs to obtain greater performance is load balancing between resources. The goal of task scheduling in CC settings is substantially enhanced by the suggested bio-inspired optimisation approach, which enhances resource utilisation while simultaneously reducing makespan and energy consumption. The simulation results indicated that there was consecutive improvement in the Makespan from 22.13s using FFO to 18.78s using LFO and Energy from 21.67 kW using FFO to 23.70 kW using LFO for task loads between 1000 to 5000 tasks. Thus, LFO was 60% efficient in energy conservation as compared to the FOA. This is a promising advance in the field of tasks scheduling optimization, LFO demonstrated good scalability, sufficient to cope with the large-scale cloud jobs, thus making its contribution to the improvement of life cycle and performance of cloud computing systems.

Nevertheless, there are characteristics that will restrict the efficacy of the method before incorporating the LFO algorithm into cloud computing task scheduling. Some of these issues include the fact that it is sensitive to population size and that it may get stuck in local optima in contexts that are very dynamic and complicated. The performance of LFO changes based on the selection of cloud infrastructure and the nature of executed tasks. Future work could act on enhancing the adaptability of the algorithm, combining it with other mechanisms or bio-inspired methods, and on applying it to real multi-cloud scenarios characterized by different resources and/or types of tasks.

References

[1] Y. Zhang and R. Yang, “Cloud computing task scheduling based on improved particle swarm optimization algorithm,” in Proceedings IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society, 2017. doi: 10.1109/IECON.2017.8217541.

[2] A. Gogineni, “Confidential Computing Architectures for Enhanced Data Security in Cloud Environments,” Int. J. Sci. Technol., vol. 16, no. 1, 2025.

[3] S. Arora and S. R. Thota, “Automated Data Quality Assessment And Enhancement For Saas Based Data Applications,” J. Emerg. Technol. Innov. Res., vol. 11, no. 6, pp. i207–i218, 2024, doi: 10.6084/m9.jetir.JETIR2406822.

[4] Pranav Khare and Abhishek, “Cloud Security Challenges: Implementing Best Practices for Secure SaaS Application Development,” Int. J. Curr. Eng. Technol., vol. 11, no. 06, 2021, doi: https://doi.org/10.14741/ijcet/v.11.6.11.

[5] N. Jafari Navimipour and F. Sharifi Milani, “Task Scheduling in the Cloud Computing Based on the Cuckoo Search Algorithm,” Int. J. Model. Optim., 2015, doi: 10.7763/ijmo.2015.v5.434.

[6] Vashudhar Sai Thokala, “Scalable Cloud Deployment and Automation for E-Commerce Platforms Using AWS, Heroku, and Ruby on Rails,” Int. J. Adv. Res. Sci. Commun. Technol., pp. 349–362, Oct. 2023, doi: 10.48175/IJARSCT-13555A.

[7] N. Patel, “Secure Access Service Edge(Sase): Evaluating The Impact Of Convereged Network Security Architectures In Cloud Computing,” J. Emerg. Technol. Innov. Res., vol. 11, no. 3, pp. 703–714, 2024.

[8] S. Murri, S. Chinta, S. Jain, and T. Adimulam, “Advancing Cloud Data Architectures: A Deep Dive into Scalability, Security, and Intelligent Data Management for Next-Generation Applications,” Well Test. J., vol. 33, no. 2, pp. 619–644, 2024.

[9] R. Tarafdar, “AI-powered cybersecurity threat detection in cloud environments,” int. j. comput. eng. technol., 2025.

[10] S. Murri, “Data Security Challenges and Solutions in Big Data Cloud Environments,” Int. J. Curr. Eng. Technol., vol. 12, no. 6, 2022, doi: https://doi.org/10.14741/ijcet/v.12.6.11.

[11] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” 2015. doi: 10.1016/j.eij.2015.07.001.

[12] T. K. K. and S. Rongala, “Implementing AI-Driven Secure Cloud Data Pipelines in Azure with Databricks,” Nanotechnol. Perceptions, vol. 20, no. 15, pp. 3063–3075, 2024, doi: https://doi.org/10.62441/nano-ntp.vi.4439.

[13] M. M. Sandhu, S. Khalifa, R. Jurdak, and M. Portmann, “Task Scheduling for Energy-Harvesting-Based IoT: A Survey and Critical Analysis,” 2021. doi: 10.1109/JIOT.2021.3086186.

[14] S. P. Godavari Modalavalasa, “Exploring Azure Security Center: A Review of Challenges and Opportunities in Cloud Security,” ESP J. Eng. Technol. Adv., 2022.

[15] V. Prajapati, “Cloud-Based Database Management : Architecture , Security , challenges and solutions,” J. Glob. Res. Electron. Commun., vol. 1, no. 1, 2025.

[16] M. S. Samarth Shah, “Deep reinforcement learning for scalable task scheduling in serverless computing,” Int. Res. J. Mod. Eng. Technol. Sci., vol. 3, no. 4, pp. 141–147, 2021.

[17] A. Gogineni, “Novel Scheduling Algorithms For Efficient Deployment Of Mapreduce Applications In Heterogeneous Computing,” Int. Res. J. Eng. Technol., vol. 4, no. 11, p. 6, 2017.

[18] M. S. R. Krishna and S. Mangalampalli, “A Systematic Review on Various Task Scheduling Algorithms in Cloud Computing,” 2024. doi: 10.4108/eetiot.4548.

[19] B. Mallikarjuna and D. Arun Kumar Reddy, “The role of load balancing algorithms in next generation of cloud computing,” J. Adv. Res. Dyn. Control Syst., 2019.

[20] A. Gogineni, “Advancing Task Scheduling in Edge Computing for Energy Efficiency: A Multi-Objective Method,” Int. J. Innov. Res. Creat. Technol., vol. 9, no. 1, 2023.

[21] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, “Load balancing techniques in cloud computing environment: A review,” 2022. doi: 10.1016/j.jksuci.2021.02.007.

[22] S. G. Domanal, R. M. R. Guddeti, and R. Buyya, “A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment,” IEEE Trans. Serv. Comput., 2020, doi: 10.1109/TSC.2017.2679738.

[23] S. Kaur and A. Verma, “An Efficient Approach to Genetic Algorithm for Task Scheduling in Cloud Computing Environment,” Int. J. Inf. Technol. Comput. Sci., 2012, doi: 10.5815/ijitcs.2012.10.09.

[24] A. I. Awad, N. A. El-Hefnawy, and H. M. Abdel-Kader, "Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments," in Procedia Computer Science, 2015. doi: 10.1016/j.procs.2015.09.064.

[25] 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.

[26] X. Chen, Z. Song, H. Zheng, and Z. Wan, "Task scheduling based on fruit fly optimization algorithm in mobile cloud computing," Int. J. Performability Eng., 2020, doi: 10.23940/ijpe.20.04.p13.618628.

[27] H. Alabdeli, S. Rafi, I. G. Naveen, D. D. Rao, and Y. Nagendar, "Photovoltaic Power Forecasting Using Support Vector Machine and Adaptive Learning Factor Ant Colony Optimization," 3rd IEEE Int. Conf. Distrib. Comput. Electr. Circuits Electron. ICDCECE 2024, pp. 2024–2025, 2024, doi: 10.1109/ICDCECE60827.2024.10549652.

[28] T. Saravanan and K. Ramesh, "A Bio-inspired Energy Efficient Dynamic Task Scheduling (BEDTS) scheme and classification for virtualization CDC," J. Eng. Res., 2024, doi: 10.1016/j.jer.2023.08.026.

[29] R. K. Kalimuthu and B. Thomas, "An effective multi-objective task scheduling and resource optimization in cloud environment using hybridized metaheuristic algorithm," J. Intell. Fuzzy Syst., 2022, doi: 10.3233/JIFS-212370.

[30] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," Knowl. Inf. Syst., 2017, doi: 10.1007/s10115-017-1044-2.

[31] M. S. R. Krishna and D. Khasim Vali, "ADWEH: A Dynamic Prioritized Workflow Task Scheduling Approach Based on the Enhanced Harris Hawk Optimization Algorithm," IEEE Access, vol. 13, no. February, pp. 1–26, 2025, doi: 10.1109/ACCESS.2025.3543880.

[32] Y. Naveen, S. B. Kasturi, C. Ramya, B. Gayathri, and S. K. Medishetti, "BACOA: Meta Heuristic Driven Hybrid Scheduling Algorithm for Improved Resource Allocation in Cloud Environment," in 2025 International Conference on Multi-Agent Systems for Collaborative Intelligence (ICMSCI), Jan. 2025, pp. 1787–1794. doi: 10.1109/ICMSCI62561.2025.10893958.

[33] P. Choppara and S. Sudheer Mangalampalli, "Resource Adaptive Automated Task Scheduling Using Deep Deterministic Policy Gradient in Fog Computing," IEEE Access, vol. 13, no. February, pp. 1–26, 2025, doi: 10.1109/ACCESS.2025.3539606.

[34] S. Kaur, J. Singh, and V. Bharti, "A Comparative Study of Optimization Based Task Scheduling in Cloud Computing Environments Using Machine Learning," in 2024 5th International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), 2024, pp. 731–740. doi: 10.1109/ICICV62344.2024.00122.

[35] S. Marathe, T. Neelima, G. Anusha, and S. K. Medishetti, "DOA: Optimizing Resource Utilization in Cloud Environment Using Multi-Objective Meta-Heuristic Scheduling Algorithm," in 2024 International Conference on Advancement in Renewable Energy and Intelligent Systems (AREIS), Dec. 2024, pp. 1–9. doi: 10.1109/AREIS62559.2024.10893648.

[36] C. R. Bennett, R. R. Oduru, D. McLaughlin, L. Parvathaneni, N. B. Riquelme, and S. K. Medishetti, "Energy and Resource Aware Scheduling in Cloud-Fog Environment using Advanced Meta Heuristic Algorithm," in 2024 2nd International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS), Oct. 2024, pp. 1422–1430. doi: 10.1109/ICSSAS64001.2024.10760415.

[37] V. M. D. Rajasingh and R. Durga, "Feasible Load Balancing for Webserver in Cloud Environment Using Energy Efficient Maximal Support Priority Scheduling Approach," in 2024 International Conference on Integrated Intelligence and Communication Systems (ICIICS), Nov. 2024, pp. 1–6. doi: 10.1109/ICIICS63763.2024.10859387.

[38] M. Ramesh, D. Chahal, C. Phalak, and R. Singhal, "Multi-Objective Workflow Scheduling to Serverless Architecture in a Multi-Cloud Environment," in Proceedings - 2023 IEEE International Conference on Cloud Engineering, IC2E 2023, 2023. doi: 10.1109/IC2E59103.2023.00027.

[39] M. Dehghani, G. Bektemyssova, Z. Montazeri, G. Shaikemelev, O. P. Malik, and G. Dhiman, "Lyrebird Optimization Algorithm: A New Bio-Inspired Metaheuristic Algorithm for Solving Optimization Problems," Biomimetics, 2023, doi: 10.3390/biomimetics8060507.

[40] A. R. Khan, "Dynamic Load Balancing in Cloud Computing: Optimized RL-Based Clustering with Multi-Objective Optimized Task Scheduling," Processes, vol. 12, no. 3, 2024, doi: 10.3390/pr12030519.