



A bio-inspired metaheuristic approach for cloud task scheduling using lateral hyena based particle swarm optimization

Meena Malik¹ · Durgesh Nandan² · Chander Prabha³ · Mueen Uddin⁴ · Biswaranjan Acharya⁵ · Yu-Chen Hu⁶ 

Received: 27 September 2023 / Revised: 16 April 2024 / Accepted: 28 May 2024

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2024

Abstract

Being a rapidly emerging field, cloud computing encourages the rapid growth of all the essential internet services. One major challenge in a cloud computing environment is balancing the load among various machines to optimize the appropriate workload, thereby providing efficient task scheduling. Various studies have been proposed and implemented for task scheduling to provide a fair load balance among all the available virtual machines on the cloud. This research work targets the optimization of virtual machines by Lateral Hyena -Particle Swarm Optimization (LH-PSO) algorithm. Our main motivation is to push parallel task scheduling proficiently, producing low response time and rapid results for all the assigned tasks. This work attempts to optimize virtual machines by proposing a Lateral Hyena-based solution and parallel task scheduling. The proposed solution provides a flexible system where virtual machines will be load-balanced by serving equal tasks. The proposed solution attempts to optimize a fitness function based on the velocity and position value of particles mapped to the characteristics of incoming requests. The practice aims to find optimized virtual machines solving tasks with an optimal solution. The performance is analyzed using the Cloudsim simulator to assess all parameters such as load distribution, average response time, memory utilization, processor utilization, and average turnaround time. The analytical conclusions represent an enhancement over the existing schemes compared to the above-mentioned parameters.

Keywords Cloud Applications · Virtual Machines · Load Balancing · Particle Swarm Optimization · Lateral Hyena-based PSO

1 Introduction

Due to cutting-edge advancements in the internet and its services, cloud computing has been developed quickly and is now inseparable from modern web and internet application design. Various kinds of Service-Level Agreements (SLAs) ensure the availability of good infrastructure and services. The cloud platform suppliers allow resources on lease to

Extended author information available on the last page of the article

support the utilization of infrastructure as per client requirements. The end users use services from service providers, which in turn get services from the infrastructure providers. As such, IT sector giants like Google, Amazon, and Microsoft have come up with cloud platforms to provide enabling services for their customers. The elementary approach of Cloud Computing is to provide necessary resources in accordance with the required services. Many requests are generated instantaneously in cloud systems from various geographically distant stations. An unmanaged scenario results in arbitrarily allocated requests to various cloud service providers, leading to overloading and underloading. This imbalanced load distribution will eventually lead to degraded overall system performance. Developing a dynamic and efficient Load Balancing (LB)scheme is very important.

To develop an efficient Virtual Machine (VM)environment, load-balancing procedures must be implemented into the system. The load balancing scheme will suggest which particular Virtual Machine will be allotted to a particular user's task, on demand, provided multiple requests may be in the queue simultaneously. To handle the system effectively, some requests must be retained in the queue, allowing the more demanding requests to be forwarded to the corresponding service providers. Therefore, the load balancing mechanism is important to decide which demand/task will continue in the queue and which will proceed for services by the providers. Mehta et al. [1] described load balancing in cloud and fog environments using a decentralized heuristic algorithm. It also takes care of Quality of Services (QoS) along with latency. Another recent case study was defined based on QoS scheduling in cloud and IoT environments [2]. It emphasizes optimized task allocation in cloud-IoT applications using VM. Jawade and Ramachandram developed a hybrid algorithm called Dragon Aided Grey Wolf optimization (DAGWO) [3] to optimize secure task scheduling in a multi-cloud environment. This work also took care of multi-objective optimization.

In [4], the authors presented an inclusive study of various load-balancing schemes and revealed all the possible advantages and drawbacks of various schemes that will direct the researchers towards subsequent improvement and developing new schemes for balancing the load. The author in [5] presented a survey of various efficient algorithms, such as 1. heuristic-centered LB algorithms (Round Robin-based algorithm, Balanced Reduce algorithm, Enhanced Max–Min based algorithm, 2. meta-heuristic-based LB algorithms (Firefly algorithm, Artificial Bee colony, Honey Bee algorithm, and more), and hybrid-centered LB algorithms.

The study [6] targeted all the recent LB schemes developed mainly to fit in the cloud systems. The classification is presented for various LB algorithms. The performance has been analyzed to evaluate the performance of all schemes on the CloudSim simulator. This scheme may be critically important for futuristic research in this domain. The suggested scheme has been implemented and executed in a real-time scenario and a systematic comparison of evaluation criteria is also presented [7].

In the same way, the article [8] concluded multiple LB schemes for cloud systems in an efficient manner as per the required taxonomy. The author concluded that the bandwidth required for the network in the cloud environment is significant and that the high traffic in the network may lead to poor use of networking resources. This may lead to some serious issues such as network failure, data loss, delay, etc. Therefore, several effective LB procedures have been developed for efficient network bandwidth usage[9]. Moreover, in [10], a hybrid LB scheme entitled Throttled and Equally Spread Current Execution has been proposed to reduce the response time, improve the provided services, reduce machine cost, and propose avoiding bottleneck issues. Similarly, the study [11] proposed a better task scheduling scheme, considering Particle Swarm Optimization (PSO) as a root algorithm.

It offers the capability to achieve effective outcomes, allowing a huge number of tasks to be scheduled simultaneously without troubling the performance and throughput of the system. Talha and Malki also introduced a new hybrid algorithm, PPTS-PSO [12], which solves the workflow scheduling among independent tasks. Mishra and Majhi gave a hybrid metaheuristic algorithm GA-JAYA [13] that was applied in healthcare data scheduling using VM for a cloud environment. However, this scheme did not effectively reflect the behavior of some important parameters, such as cost or energy consumption. Furthermore, the proposed algorithm must be improved to attain a synchronized workflow while scheduling tasks [14, 15]. Therefore, our proposed work aims to optimize the VM using Lateral Hyena and implement PSO for load balance among VMs, thereby accomplishing concurrent task scheduling.

Several authors propose several techniques [15] that employ PSO and other nature-inspired techniques for load balancing through proper scheduling in the cloud environment. However, the load balancing as a typical scenario of modern cloud applications, which supports both push and pull-based operations, has a complex implementation. This research is primarily focused on the scenario of web-based pull requests. The IoT and sensor applications of the cloud have a different approach, as the devices attached may receive push notifications when certain events are triggered.

- The main contributions of the proposed work are presented below.
- Optimization of scheduling of VMs using the proposed scheme, which associates the Lateral Hyena approach with Particle Swarm Optimization (LH-PSO).
- Targeting the proposed scheduling scheme for real-time load distribution.
- Evaluate performance using parameters like average response time, average load, average turnaround time, memory utilization, runtime, and CPU utilization to measure its effectiveness.

This paper is organized as follows: Sect. 1 –presents an understanding of all the fundamental concepts of Virtual Machine, Cloud Computing, Load Balancing, and Task scheduling in the cloud scenarios. Section 2 presents an extensive review and discusses the analysis of existing schemes for balancing the load in cloud systems. Section 3 outlines the proposed methodology and presents the hybrid scheme to achieve concurrent Load Balancing. Section 4 discusses the outcome obtained and compares the existing schemes by implementing the proposed scheme. The complete study and work have been summarized in Sect. 5.

2 Related work

Researchers have worked on numerous approaches and schemes for balancing the load among nodes in cloud environments. The author in [16] proposed a scheme to distribute load dynamically using a hybrid approach that regulates the velocity of MPSO (Modified Particle Swarm Optimization) combined with a Q-learning algorithm based on gbest and pbest. The scheme managed to balance the load for independent tasks, improving throughput, system performance, makespan, and reducing waiting time. Further, this study also suggested dynamic load balance for the dependent task in cloud environments. A PSO-based scheme in [17] offered a static method for scheduling the independent and non-pre-emptive tasks based on the PSO algorithm. Here, the Aiding LB factor optimizes

the performance of basic PSO, and resource utilization is improved by 22%. A resource scheduling scheme in [18] integrated load balancing and task scheduling by implementing a fuzzy-based resource scheduling to obtain efficiency and load optimization in the cloud computing area. The scheme intends to be effective, providing a 7% rise in resource scheduling and 33% in response time. It applies response time and success rate factors to schedule privacy-aware resources.

Furthermore, the authors in [19] suggested a resource-aware scheme in the domain of green computing. As we all are aware, green computing areas are facing issues related to low performance and high energy costs. A resource Aware LB clonal algorithm has been proposed to resolve the above-mentioned issues that perform effectively, thereby enhancing the balance as well. The simulation results demonstrate an effective reduction of energy consumption in green computing. Further, it also proposes the application of energy efficiency schemes for big data centers. The scheme in [20] used genetic algorithms for optimization, combined a greedy approach with a modified genetic algorithm, and used a smaller number of iterations for task scheduling. The scheme performed well and suggested a stochastic-based approach for improving cloud resources.

A capacity-based scheme termed CPDALB in [21] offered load balancing by capacity-focused deadline parameters that mainly emphasize a small cost to satisfy the customer. It ignored workflow-based applications and Quality of Service (QoS) parameters. The results indicated that the proposed scheme outperforms all the competitive algorithms. In [22], the authors suggested a genetic algorithm that uses queuing theory for optimized service selection and a heuristic approach for load balancing. The computational outcomes represent the scheme's effectiveness, but the model fails to represent continuous orders and service failure while servicing and assumes all the tasks to be non-preemptive. The study in [23] pointed to task scheduling as a primary challenge for QoS and cloud optimization, proposing two new hybrid heuristic algorithms. One is fuzzy-based PSO, and the other is a combination of simulated annealing with PSO. The experimentation shows clear and effective results for fuzzy-based PSO in terms of makespan, queue length, waiting time and resource utilization. Further, this study suggested improving the work for the robustness of the proposed scheme by focusing on several QoS parameters. Authors in [24] proposed a combination of the firefly algorithm, which can efficiently perform the minimization of search space, with IMPSO, that improves response time, resulting in the FIMPSO algorithm. The hybrid version effectively improved memory utilization, CPU utilization, throughput, and reliability of the full system.

In [25], the paper presented a systematic review of all the algorithms used for context-aware scheduling in cloud environments. Due to its distributed and dynamic nature, different authors use hybrid or modified PSO to obtain optimum results. The comparison table clearly represents PSO as an efficient approach over the genetic algorithm due to its simple concept, better results, fast convergence, and easy programmability. The authors in [26] suggested a task scheduling algorithm offers stochastic development by recognizing two essential factors, virtual machine selection and task scheduling, as the foremost concern for optimizing different processes. It uses gravitation search for solving optimization and NSGA for exploration. The results show low cost, low power consumption and lesser response time via combination of GSA and NSGA. The study in [27] proposed clustering on the basis of resource provisioning and allocation on the basis of the OCRP optimal cloud resource provisioning algorithm. The scheme resulted in improved cost, execution time, and low memory utilization compared to existing schemes. However, gaps are present that need to be proved and clarified to confirm efficiency. The authors proposed the PSO-ALBA algorithm [28], an

adaptive load-balancing approach applied with PSO to enhance the performance in both overloaded and underloaded conditions. The proposed algorithm performed better than the existing heuristic scheduling algorithms.

In [29], Lawanyashri et al. proposed a multi-objective hybrid fruitfly optimization technique based on SA for load balancing in cloud computing environments. The authors carried out two separate experiments. The original FOA and Hybrid FOA were compared in the first, while the second FOA-SA-LB was compared with PSO, HBB-LB, and EFOA-LB. The proposed method achieves greater performance than existing ones. For load balancing, the authors in [30] have combined evolutionary algorithms and fuzzy scheduling practices to enhance the resource allocation to maintain the LB in cloud providers. The proposed technique selects the most suitable host based on the number of VMs. Further, the tasks are assigned among VMs while maintaining load balance. The authors suggested the use of bounding procedures to further enhance load balancing in the future. The article [31] suggested a wolf-based optimization for cloud applications to efficient task scheduling and load balancing thereby achieving lesser runtime and response time. The author of article [36] considered a novel scheme for task scheduling in cloud environments termed Whale Optimization Algorithm (WOA) that aims to improve the cloud system performance with the available computing resources, resulting in a new proposed scheme termed Improved WOA. The results indicate better accuracy, convergence speed, and resource utilization for small- and large-scale tasks.

The original PSO technique is presented in [32], which gives a baseline for the application of PSO in real-time scenarios involving load balancing / other issues in the cloud and similar environments. Nowadays, IT industries are experiencing a rising trend for cloud computing, and the requirement for storage, resources, and infrastructure is very high. Along with all these challenges encountered by Cloud applications, other key challenges are task scheduling and load balancing. Load balancing ensures even workload distribution among all the nodes to improve resource utilization and system performance. So, the present work tries to optimize the system by distributing a uniform load among all the VMs using Lateral Hyena and PSO, implementing concurrent tasks, and minimizing the overloading and under-loading of VMs.

3 The proposed methodology

The paper suggests a hybrid approach that tries to identify the optimal Virtual machine with an average, assigned load to avoid load imbalance among nodes. The main objectives are loading balance and effective task scheduling by implementing the proposed novel scheme. It can be achieved by following the steps mentioned in Fig. 1.

At the initial stage, the generated request or task is submitted to the queue assigned for a particular virtual machine and to the VM manager. Subsequently, VMs will be created, followed by uploading the tasks for a particular VM. To start optimization for the PSO algorithm, all the basic parameters need to be initialized along with the initialization of all the particles by their corresponding position value and velocity value for solving a target problem. LH operation needs to be applied to update the fitness value, where the population will be initialized to evaluate the fitness value. A lateral method will be applied to update the fitness value of each searching agent, and the best fitness value will be recorded. The particle with the best value for velocity and position will serve as a reference for the optimal solution. So, the suggested method, Lateral Hyena-based optimization, optimizes extremely loaded VMs

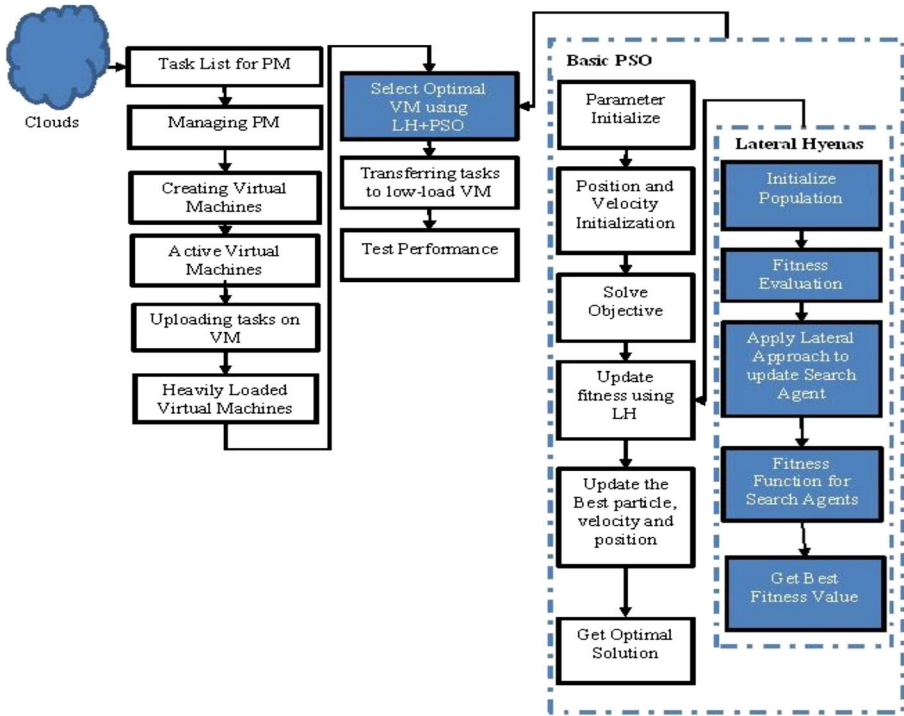


Fig. 1 Flowchart of the proposed algorithm

by transmitting the tasks to reduced-load VMs. To examine the effectiveness, the scheme is evaluated for several performance factors, such as memory utilization, CPU utilization, average turnaround time, average load, completion/ runtime, and average response time.

PSO is a meta-heuristic-based optimization methodology that aims to improve a problem using repeated iterations. It aims to obtain a contender solution to the identified issue and maintain a reference quality measure. It resolves the identified problem by retaining and recognizing a candidate solution for the object population (particles) and flow is represented in Fig. 2. Further, there is a need to move the object particles/population around the identified Search Space (SS) based on a defined analytical procedure. The velocity and position are major constraints for the planned analytical method. The change in value for a single particle represents its best-reported position and leads to the best position in SS. The best value shows the way to update for an optimal position, as an outcome, will be accountable for moving the complete swarm near the optimal solutions. This algorithm doesn't exploit the gradient, which is focused on optimization. So, there is no need to diversify the optimization problem, which is mandatory in some standard optimization schemes such as gradient descent and quasi-newton methods. This work resolves major cloud issues such as TS and LB and obtains an optimal solution.

Firstly, Eqs. (1) and (2) decide the initialization of velocity and position of the particles of a swarm [27].

$$\text{Position} : Y_i = (y_{i,1}, y_{i,2}, \dots, \dots, y_{i,n}) \in \mathbb{R}^n \tag{1}$$

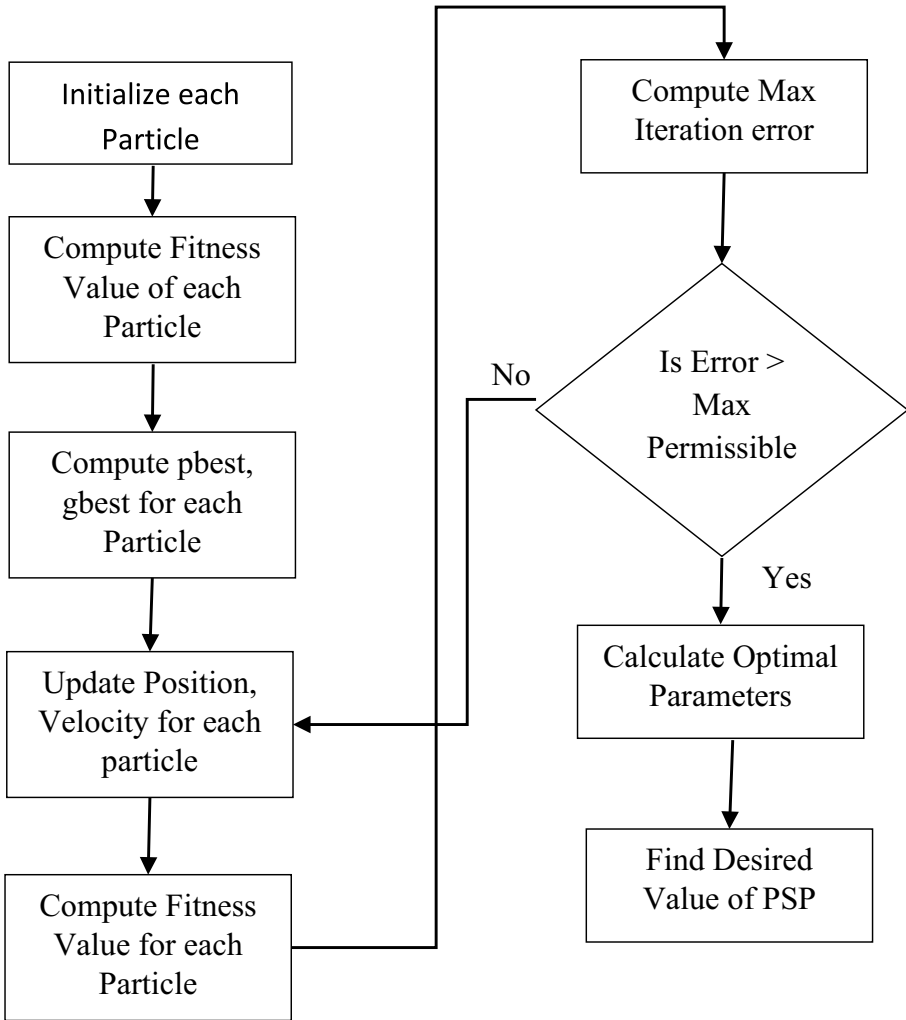


Fig. 2 Flowchart of the PSO algorithm

$$\text{Velocity} : V_i = (v_{i,1}, v_{i,2}, \dots, v_{i,n}) \in \mathbb{R}^n \tag{2}$$

$$V_i(j + 1) = \text{Inertia} + \text{cognitive} + \text{social} \tag{3}$$

The component Inertia refers to the “force” component, which is applicable to all particles in the PSO environment. The term “social” refers to the social hierarchy of spotted hyenas and the term cognitive refers to the rate at which the parameters’ values are adjusted to get optimum values. This is in conjunction with the term “learning rate” used by most deep learning algorithms based on neural networks.

$$V_i(j + 1) = \omega \times V_i(j) + k1 \times \text{random1} \times _PB\text{Best}_i - y_i(j) + k2 \times \text{random2} \times _GB\text{Best}_i - y_i(j) \tag{4}$$

Here, ω , k_1 , k_2 represent the constants and rand_1 , rand_2 indicate the random variable. For further updating, the velocity and position of the particles, Eqs. 5 and 6 can be used.

$$Y_i(j+1) = Y_i(j) + V_i(j+1) \quad (5)$$

$$Y_i(j+1) = \text{Inertia} + \text{cognitive} + \text{social} \quad (6)$$

Algorithm 1 Explains basic PSO.

```

Basic-PSO( )computes Particle Velocity and Position.
1: For each Particle of the Swarm//for all dimensions
    Initialization of Particle Position  $Y_i$ , Velocity  $V_i$  //Set PSO parameters
End
For each incoming VM request:
2: For each Particle of the Swarm //for iteration 1 to max
    Perform Velocity Reset
    Perform Position Update
3: Compute Fitness Value (FV) //for 1 to Population size
    If  $\text{fitness}(Y_i) > \text{Fitness}(pbest_i)$ 
        Update  $pbest_i = Y_i$ 
    Select best Fitness Value // for 1 to Population size
     $Pbest = \text{optimal FV}$ 
4: For each particle //for 1 to population size
    If  $pbest < gbest$ 
        Update  $pbest$  to  $gbest$ .
End
5: For each particle
    Compute Particle Velocity and Position
    Update Particle Velocity and Position
6: While Max Iteration or Min Error.

```

The algorithm 1 explains the basic PSO process and helps compute optimal particle velocity and position. In Fig. 2, the flowchart for PSO represents the process more clearly. Initially, the population is initialized by assigning values to all the particles. The next step is to calculate the Fitness Value (FV) for all the particles and determine an improved response compared to the optimal value previously. So that the newly improved value can be set as the next reference for FV. The particles possessing the best fitness value will be selected, and the velocity for selected particles needs to be calculated and updated. All the above-mentioned steps will be repeated till the mentioned criteria are met.

3.1 Lateral hyena

The LH approach follows an iterative way for optimization problems. It is inspired by the hunting strategy and management hierarchy used by spotted Hyena, which attacks big prey in packs. It discovers optimal solutions by exploring search space in a similar way to a pack of Hyenas. The algorithm represents four classes of hyenas: spotted, striped, brown, and ardwolf. Their classification process is implemented to pretend the leadership/management hierarchy. Three important steps for hunting are mainly applied to derive optimization. The steps are mentioned as searching for the prey initially, then encircling the finding, and then attacking the prey. The proposed scheme LH evolved to be an efficient meta-heuristic optimization algorithm. It shows the capability to resolve all the unimodal problems, proves effective results for even complex functions, and is easily able to avoid local minima as well. It claims a higher exploration ability for all the identified multi-modal issues.

In addition, the analytical model for the encircling process for hunting can be represented by Eqs. 7 and 8.

$$x = |ky_p(t) - y(t)| \quad (7)$$

$$y(t + 1) = y_p(t) - ax \quad (8)$$

Here, t indicates iteration number, a and k are coefficient vectors, y_p is the position vector of the prey and y is LH's position vector.

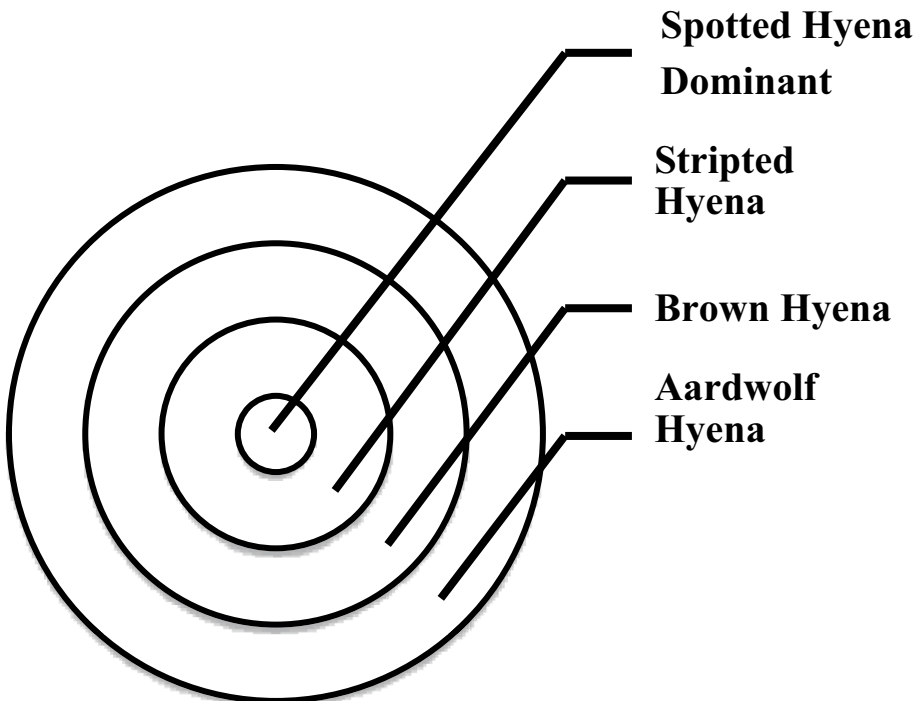


Fig. 3 Social dominance hierarchy in Hyenas

$$A = 2 * f * r_1 - a \quad (9)$$

$$C = 2 * r_2 \quad (10)$$

Here, r_1 , r_2 , are random vectors that have values from (0,1), and f is decreased linearly from value (2,0).

$$d_\alpha = |k_1 y_\alpha(t) - y|, d_\beta = |k_2 y_\beta(t) - y|, d_\delta = |k_3 y_\delta(t) - y| \quad (11)$$

$$y_1 = y_\alpha - a_1(d_\alpha), y_2 = y_\beta - a_2(d_\beta), y_3 = y_\delta - a_3(d_\delta) \quad (12)$$

$$y(t+1) = (y_1 + y_2 + y_3)/3 \quad (13)$$

Figure 3 illustrates that spotted Hynas is the most dominant class of Hyena in the entire clan. These are followed by striped, who Brown and Aardwolf Hyenas in turn follow. It presents the leadership and management hierarchy that is followed while catching the prey and securing the clan and its movement. This hierarchy is mapped to the optimization problem's characteristics and dealt with how the hyenas deal with the scenario.

Algorithm 2 Explains the process of Lateral Hyena.

Lateral Hyenas

```

1: Start
2. For each agent of Swarm
    a. Initialization of population of Hyenas. //for 1 to n all hyenas of swarm
    b. Calculate fitness value for all the grading agents and search agents
3. Choose the best FV for  $Y_a$  to represent the optimal solution,  $Y_b$  next optimal search agent and  $Y_c$  third
    optimal.
4. Set  $t_I=0$ 
    While ( $t_I < MAX\_Iteration\_Count$ )
        For each search agent
            Find and assign Current_Position.
        End
4: Revise  $b$ ,  $B$ ,  $C$  in a lateral manner
    For each search agent
        Compute FV and assign  $Y_a$ ,  $Y_b$ ,  $Y_c$ 
        Increment  $t_I$ 
    End
End

```

Algorithm 2, evidently evidently signifies the Lateral Hyena process. The input parameters considered are *Problem_Size* and *Population_Size* for the algorithm. Firstly, the population initialization is completed by following the spotted Hyena. The next step evaluates

Table 1 The initial parameters used in the proposed scheme

Sr. No	Parameter Name	Values
1	Number of processors	2
2	Max iteration	100
3	Training factors	3
4	Higher and lower inertia weight	Higher=0.5, Lower=0.10
5	Storage capacity	1,000,000
6	Input size (MI)	Depends on the task size
7	Bandwidth, Mbit/s	3000
8	Output size (MI)	Depends on the Input size

Table 2 Analysis of runtime of the proposed scheme vs. the existing systems

No of Tasks	100	150	200	250	300	400
HBB-LB	29.5	61.5	71.9	85.5	95.5	110.5
EBCA-LB	78.1	99.01	62.01	165.2	193.02	221
Fuzzy-Based	24.5	52	62.75	77.0	85.75	96.5
LH-PSO (Proposed)	21.25	46.59	58.06	71.65	75.39	90.57

the fitness value *FV* for all search and grading agents. It selects the first optimal solution, second best, and third best solution for the search agents. Afterward, each search agent’s corresponding position needs to be revised until it reports the timing parameter, *t1* is less than *Max_Iteration*. After updating the position value for all the search agents, the value of *Pg1_best* will be revised by the best *FV*.

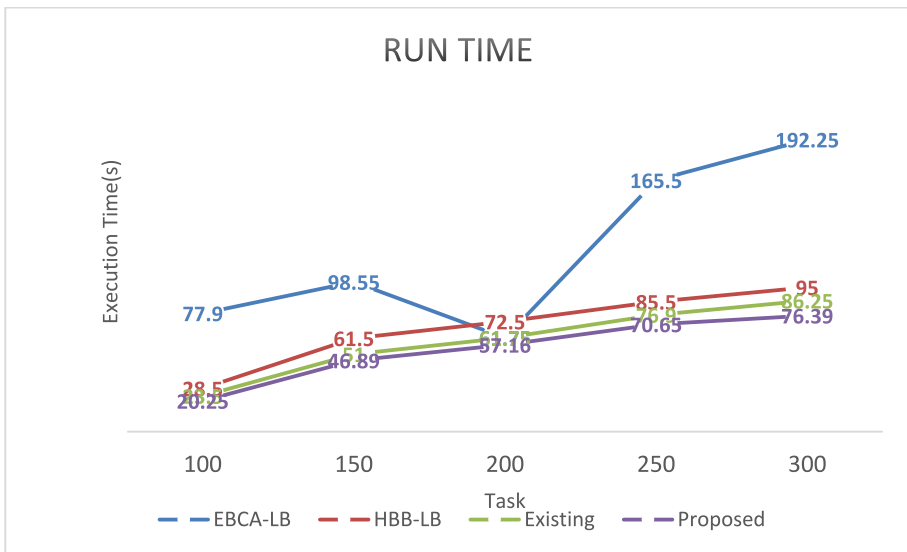


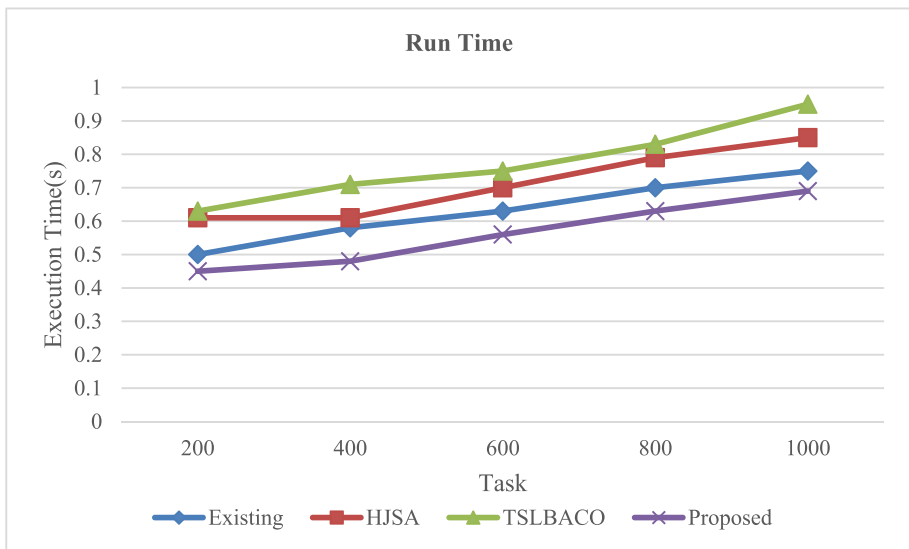
Fig. 4 The runtime of the proposed scheme vs. the existing systems

Table 3 The runtime of the LH-PSO vs. the existing systems

No of Task	200	400	600	800	1000
Existing Approach	0.51	0.59	0.64	0.71	0.75
HJSA Approach	0.62	0.61	0.72	0.78	0.86
TSLBACO Approach	0.64	0.72	0.74	0.84	0.96
Proposed Approach	0.46	0.48	0.55	0.64	0.68

3.2 Lateral hyena based-PSO

This section represents the hybrid approach of Lateral Hyena with the PSO algorithm to solve load-balancing issues in the cloud domain. All the advantages of the PSO approach, such as a convenient model, simple implementation, robustness for parameter control, and

**Fig. 5** Runtime of the proposed scheme vs. the existing systems**Table 4** Average response time performance of LH-PSO vs. the existing methods

Methods	Avg Load (ms)	Avg turnaround time (ms)	Avg response time (ms)
FCFS	0.45	41.77	30.74
RR	0.44	41.09	30.4
SJF	0.485	41.55	30.23
IPSO	0.447	57.64	49.33
Firefly	0.472	55.53	48.88
FIMPSO	0.246	21.08	13.59
FF-IPSO	0.258	22.14	15.22
Proposed	0.232	20.57	12.97

efficient computations, mark it better than earlier methodologies. So, this work suggests PSOs find optimal VMs for scheduling tasks to balance load distribution into the system. There is a strong requirement to dodge the probability of local minimum problems due to PSO. The issue of LM can be solved or minimized by applying the Lateral Hyena scheme along with PSO. LH performs very effectively when combined with PSO for task scheduling and load balancing in cloud areas.

LH employs the scheme by moving some particles to certain positions by following adaptive mode instead of random positions. This enhancement is implemented by LH to overcome issues of PSO. The complete process is well defined in steps in Algorithm 3.

Moreover, the LH algorithm may apply task scheduling in parallel with LB, by consumption of the least possible memory, simple implementation procedures, and more rapid convergence ability that illustrates its efficiency for execution in cloud environments.

To calculate the fitness value, LH-PSO uses the following computations for position and velocity update:

$$\text{Updating Position : } Y_i(j+1) = X_i(j) + V_i(j+1) + d_\alpha = \left| k_{hy} y_{hi}(t) - y \right|$$

$$\text{Updating Velocity : } y_i(j+1) = \text{Inertia} + \text{cognitive} + \text{social} (t+1) \\ + (y_1 + y_2 + y_3 \dots \dots + y_n) / n$$

where, $n \rightarrow$ number, $h \rightarrow$ hyena.

Algorithm 3 represents hybrid Lateral Hyena-PSO process.

Lateral Hyena-PSO (Pop_Size ,poss)

```

1: Initialize the full population of particles
for I = 1 to Max_iter do
    for j= 1 to Pop_Size do
        Execute PSO
        Update particle position and velocity.
2: if rand(0,1) < poss
    Set b, B and C
    For g = 1 to max_ iterations_1 do
        For n = 1 to max_ iterations_2 do
            Run LH
3: Update the  $\alpha$  ,  $\beta$  ,  $\gamma$  hyenas' position laterally
    Update b, B, C
    End for
    End for
4: Position of present particle = mean of three optimal hyena' position
    End if
    End for
    End for

```

Where,

<i>Pop_Size</i>	Population Size.
<i>Max_Iter</i>	Maximum Iteration.
<i>Poss</i>	Minimum Rate of Possibility.
<i>max_terations_1/2</i>	Threshold value for stabilization using try and error.

4 Results and discussion

The offered scheme LH-PSO offers a hybrid model for load balancing to achieve efficiency. This segment represents all the results obtained by implementing Lateral Hyena-PSO. Cloudsim simulator has been used to analyze the system's effectiveness. The proposed scheme is compared with the already existing scheme considering average runtime, response time, average load, memory utilization, CPU utilization, and average turnaround time. The initial parameters used for the proposed scheme are presented in Table 1.

The efficiency of the proposed method is evaluated in terms of runtime by comparing it with existing methods, such as HBB-LB and EBCA-LB. The results are represented in Table 2.

The task length and speed of VMs are important constraints for assigning various tasks to the resources. Here, multiple task assignments are started. When 100 tasks are assigned, EBCA-LB spends 78.1 s, HBB-LB spends 29.5 s, and the fuzzy-based scheme uses 24.5 s. However, the proposed scheme runtime is calculated at 21.25 s for the same count of tasks. Correspondingly, for different numbers of task assignments, such as 100, 150, 200, 250, 300, and 400 tasks, the proposed scheme manages lower runtime than the existing schemes [33]. The graph representation for runtime is in Fig. 4.

It points out that the proposed system achieves a lower runtime than the existing one in terms of increasing the number of assigned tasks. Additionally, the suggested scheme is investigated similarly by comparing it with existing approaches, such as HJSA and TSLBACO, concerning runtime evaluation to test efficiency. The results are presented in Table 3.

The number of tasks assigned is increased further to all the VMs. The results indicate that LH-PSO outperforms HJSA and TSLBACO and is a more efficient algorithm for finding VM optimization. For 200 tasks, the existing method takes 0.51 s, HJSA takes 0.62 s, and TSLBACO takes 0.64 s. Conversely, our proposed scheme completes in 0.46 s only. Similarly, our proposed scheme confirms the lesser amount of runtime compared to 400, 600, 800, and 1000 tasks. Typically, increasing the number of tasks results in higher running time. However, our proposed scheme manages the lowest time against the increase in task assignment. The graphical representation is in Fig. 5.

The results plotted for comparative analysis clearly show the proposed scheme reduces runtime for task execution over already existing methods. The computational effectiveness of PSO and parallelism for load balancing by LH method enable the suggested scheme to be efficient enough to minimize overall runtime for various tasks.

The other parameters considered for assessing the proposed scheme are response time, average load, CPU utilization, average turnaround time, and memory utilization. The results are presented in Table 4. It can be observed that the RR technique confirms 0.44 s as the average load, 41.09 ms for the average turnaround time, and the average response time of 30.4 s. In the same way, the existing methods SJF, RR, FCFS, IPFO,

Firefly, FF-IPSO and FIMPSON confirm higher values for all the considered parameters and are represented graphically in the under-mentioned figures.

Figure 6 clearly displays that the suggested scheme LH-PSO confirms minimum average load compared to the present schemes such as FCFS, RR, SJF, IPSO, Firefly, FF-IPSO and FIMPSON. Correspondingly, the average turnaround time is plotted against all the existing schemes in Fig. 7.

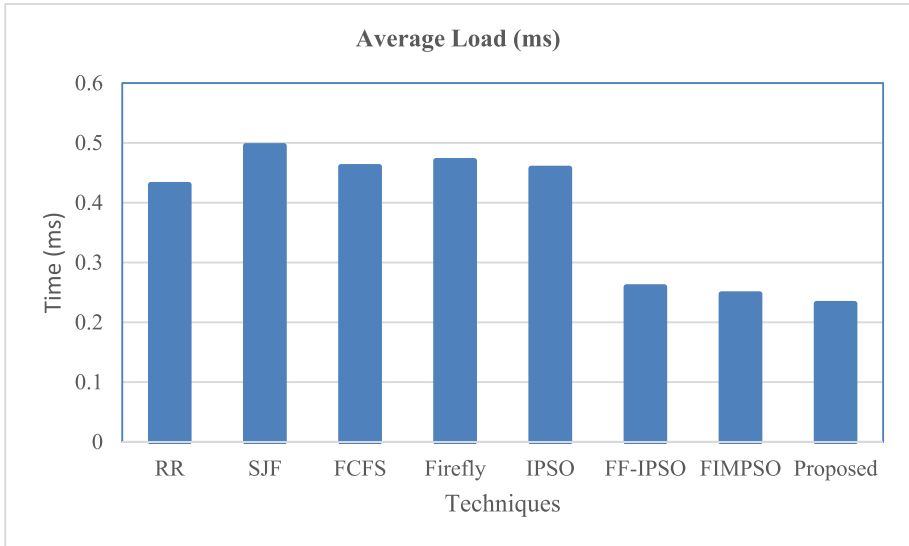


Fig. 6 Average load of LH-PSO vs. the existing systems

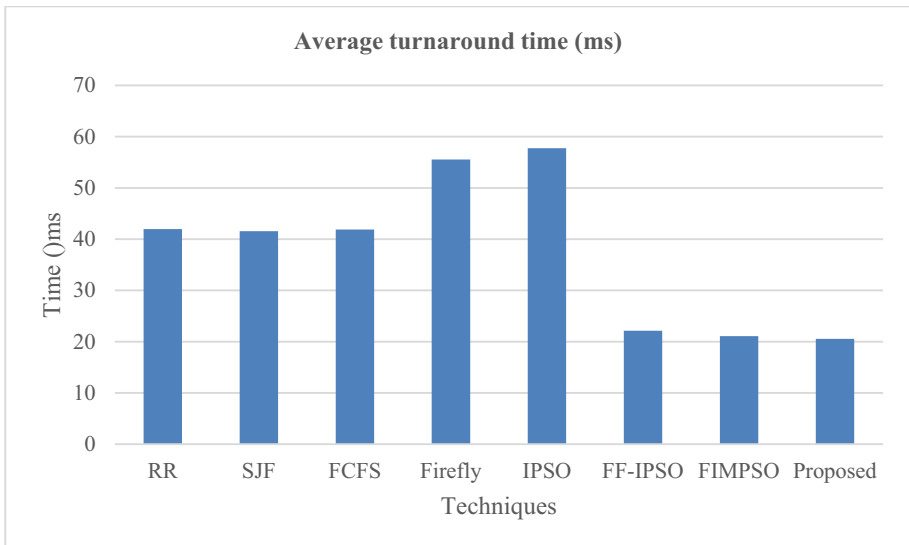


Fig. 7 Average turnaround time of LH-PSO vs. the existing methods

In Fig. 7, it can be observed that the suggested scheme confirms a lower turnaround time than the existing methods. In addition, the calculated average response time of the proposed scheme is compared using the mentioned schemes, which proves its efficacy for the proposed scheme only. The response time is a prominent aspect of cloud applications. The corresponding results are represented in Fig. 8.

The suggested system responds very fast to all the user's requests, thus demonstrating the system's proficiency in response time. From Fig. 8, it can be easily stated that the response time is minimized compared to the traditional approaches. The markable performance of the proposed system specifies its effectiveness. Normally, hybrid systems result in slow response. However, the proposed system has proved to have higher computational efficiency. This makes them work faster. The proposed scheme LH-PSO is evaluated for memory and CPU utilization for different tasks. The evaluation is represented in Table 5.

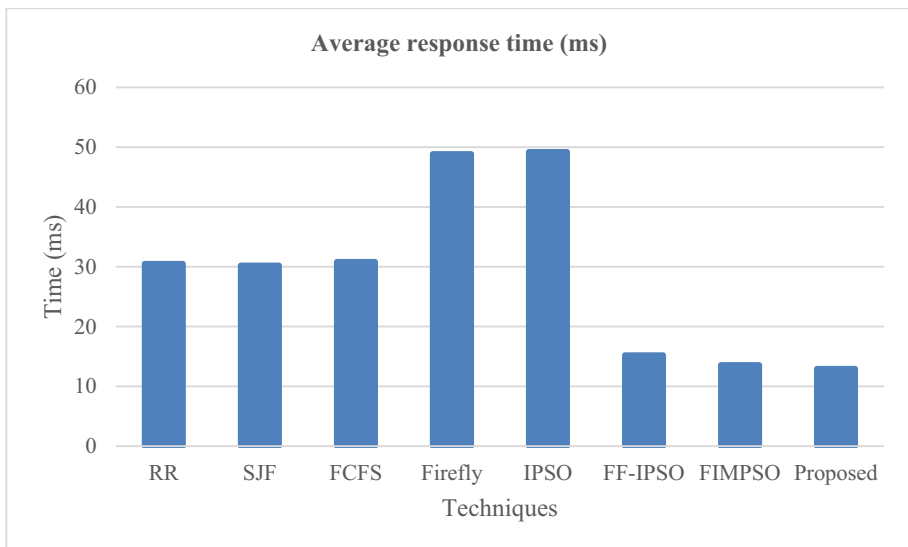


Fig. 8 Average response time of LH-PSO vs. the existing schemes

Table 5 CPU utilization of the proposed scheme vs. the existing methods

Approaches	Task Size			
	Small	Medium	Large	Extra Large
RD	46	51	65	71
WRR	49	58	68	75
DLB	51	64	70	81
LB-BC	58	68	76	85
LB-RC	65	76	86	90
IPSO-Firefly	68	78	89	97
FIMPSO	71	79	95	99
Proposed	75	83	99	100

Different categories of tasks, such as extra-large, large, medium and small, are considered. The existing RD method used CPU up to a slight extent for all categories of tasks. Similarly, the other methods considered, i.e., WRR, DLB, LB-BC, LB-RC, IPSO-Firefly and FIMPSO, result in poor CPU utilization compared to the proposed scheme. It is examined by experimental analysis that the proposed scheme utilized CPU very efficiently at a level of 75% corresponding to small tasks, 83% for medium-level tasks, 99% for large-sized tasks, and 100% for extra-large-sized tasks. The graphical representation is shown below in Fig. 9.

By analyzing Fig. 8, it can be easily identified that the proposed system outperforms all the existing methods in terms of CPU utilization. Increasing the number of tasks typically affects the CPU utilization in terms of efficiency. But, the proposed LH-PSO scheme proves its ability to effectively perform CPU utilization, making it suitable in cloud systems for load balancing. The proposed scheme is further analyzed for memory usage and represented in Table 6.

Also, the proposed scheme finds its effectiveness for memory utilization in comparison to the traditional algorithms. Different categories of tasks are considered for comparison, such as extra-large, large, medium, and small. The methodical results reveal the effectiveness of the proposed scheme by showing small task utilization as 62%, medium task utilization as 76%, large task utilization as 85%, and 91% extra-large tasks. We can determine the high efficiency of the proposed scheme compared to the existing methods represented in Table 6. This means in Fig. 10 that the proposed scheme utilizes memory very efficiently in terms of memory for all the task categories.

The proposed scheme represents efficiency in terms of memory utilization compared to the existing systems. As the number of tasks increases, the effectiveness of the memory utilization remains balanced for the offered scheme over the existing schemes. It validates that the proposed scheme is more reliable than the existing scheme, even for increasing the number of tasks assigned to the VMs. It expresses the capability to identify the optimized

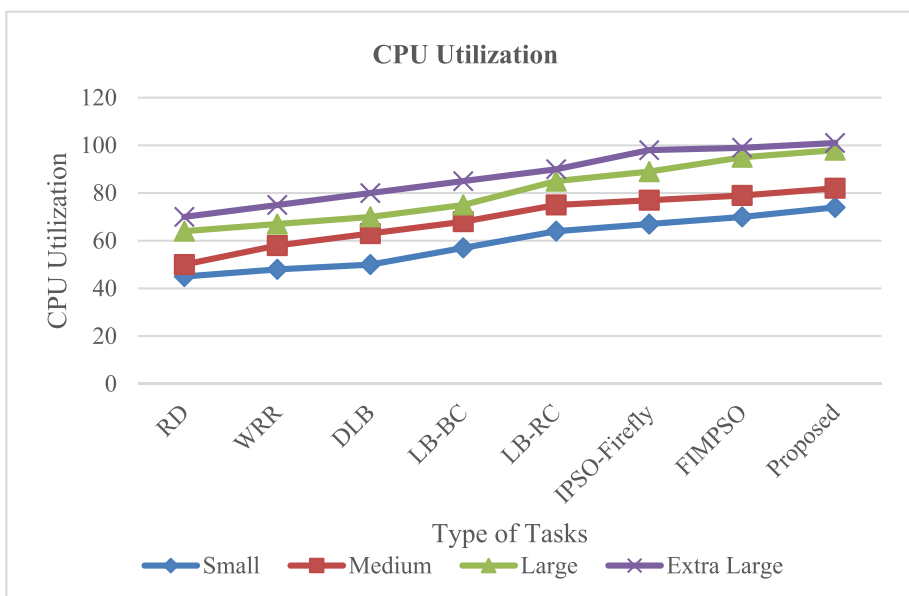
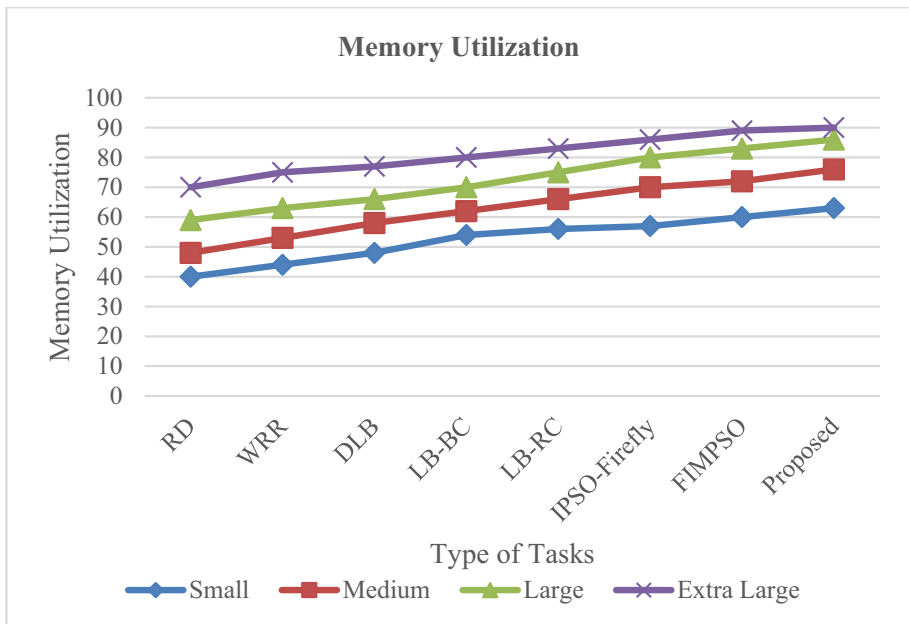


Fig. 9 CPU utilization of the LH-PSO vs. the traditional methods

Table 6 Memory utilization of the proposed scheme vs. the existing systems

Approaches	Task Size			
	Small	Medium	Large	Extra Large
RD	41	48	58	72
WRR	45	54	63	75
DLB	48	58	67	78
LB-BC	55	63	70	80
LB-RC	56	65	76	84
IPSO-Firefly	58	70	80	86
FIMPSO	61	73	84	88
Proposed	62	76	85	91

VM, perform efficient LB, and provide a fast response to user requests. The work in [34] implements a hybrid scheme termed ant colony and lion optimization designed to resolve task scheduling problems in cloud applications. However, the work represents improved results by considering fewer parameters such as makespan time, imbalance degree, and response time. The limited number of parameters results in unsatisfactory system performance efficiency. The present work reflected several major constraints for an investigation, like average load, runtime, memory utilization, turnaround time, response time, and CPU utilization. The methodical outcomes represent better performance and are more efficient for our proposed scheme concerning all the parameters. It is analyzed from the experimental results that Lateral Hyena-based-PSO is reliable and suitable for resolving task scheduling issues in cloud applications.

**Fig. 10** Memory Utilization of LH-PSO and traditional methods

Several benchmark papers that used such metaheuristic approaches discuss the overhead induced by such algorithms. A number of such papers are also discussed in the literature review section. In most cases, after the algorithm has been initialized for parameters when the new request arrives, which prompts the cloud system for algorithm execution, the algorithm usually converges in a small number of iterations. This results in minimal time complexity overhead once the scheduler starts execution.

The proposed algorithms can be used efficiently to process complex analytics in bulky datasets, and speed can be accelerated with the help of task scheduling in various computing frameworks. Secondly, it can be used in Scientific calculations/simulations where intensive computation is required. With the help of the proposed task scheduling and load balancing scheme, these tasks can effectively decrease the execution time drastically. In the healthcare industry, it may be used to optimize various tasks such as patients' treatment plans, analysis of imaging data, and medical diagnoses. In the e-commerce industry, the solution may be used effectively to optimize all possible pricing schemes, improve customer retention, and improve personal recommendations.

5 Conclusion

This work proposes a Lateral Hyena-based PSO algorithm, which recommends improvements to address the basic limitations of the PSO algorithm for scheduling various tasks in cloud scenarios. For the successful working of lateral strategy, the optimization between the exploration phase and exploitation phase plays a very important role. The Lateral schemes can efficiently regulate all the control parameters as per the complexity by instantly calculating the fitness value on demand. The scheme calculates fitness convergence standards, which provide efficient optimization by announcing a threshold for stopping the optimization process. The results outline that the proposed scheme is effective compared to the existing methods by reducing the system's average load, turnaround time, and response time. The offered scheme efficiently uses CPU and memory as compared to the traditional ones.

As a future perspective of the current work, research will be directed towards cloud resources based on the user requirement patterns, which can be deduced using a time-series approach based on the hourly, weekly, and monthly usage patterns across data of a considerable number of years of usage. Also, an investigation will be conducted on the suitability of the variants of PSO in accordance with the variance in the size of incoming requests, as suggested in [35]. This is because most of the variants of PSO techniques, in addition to the proposed algorithm, work best under low variance.

Funding The authors declare that they do not have any funding or grant for the manuscript.

Data availability The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Declarations

Ethical approval No animals were involved in this study. All applicable international, national, and/or institutional guidelines for the care and use of animals were followed.

Conflict of interest The authors declare that they do not have any conflict of interests that influence the work reported in this paper.

References

1. Mehta R, Sahni J, Khanna K (2023) Task scheduling for improved response time of latency sensitive applications in fog integrated cloud environment. *Multimed Tools Appl* 82(21):32305–32328
2. Chowdhary SK, rao ALN (2023) A task clustering based QoS aware scheduling algorithm for task execution in cloud-Iot model for education services. *Multimed Tools Appl* 82(29):44783–44800
3. Jawade PB, Ramachandram S (2023) DAGWO based secure task scheduling in Multi-Cloud environment with risk probability. *Multimed Tools Appl* 83(1):2527–2550
4. Nuaimi K, Mohamed N, Alnuaimi M, Al-Jaroodi J (2012) A Survey of Load Balancing in Cloud Computing: challenges and algorithms. *Proceedings - IEEE 2nd Symposium on Network Cloud Computing and Applications*. NCCA 2012:137–142. <https://doi.org/10.1109/NCCA.2012.29>
5. Mishra SK, Sahoo B, Parida PP (2020) Load balancing in cloud computing: a big picture. *J King Saud Univ-Comput Inf Sci* 32(2):149–158. <https://doi.org/10.1016/j.jksuci.2018.01.003>
6. Hota A, Mohapatra S, Mohanty S (2019) Survey of different load balancing approach-based algorithms in cloud computing: a comprehensive review. In *Computational intelligence in data mining: Proceedings of the international conference on CIDM 2017* (pp 99–110) Springer Singapore
7. Gabi D, Ismail AS, Zainal A, Zakaria Z (2017) Solving task scheduling problem in cloud computing environment using orthogonal taguchi-cat algorithm. *Int J Electr Comput Eng* 7(3):2088–8708
8. Gamal M, Rizk R, Mahdi H, Elnaghi BE (2019) Osmotic bio-inspired load balancing algorithm in cloud computing. *IEEE Access* 7:42735–42744. <https://doi.org/10.1109/ACCESS.2019.2907615>
9. Thakur A, Goraya MS (2017) A taxonomic survey on load balancing in cloud. *J Netw Comput Appl* 98:43–57. <https://doi.org/10.1016/j.jnca.2017.08.020>
10. Shafiq DA, Jhanjhi NZ, Abdullah A, Alzain MA (2021) A load balancing algorithm for the data centers to optimize cloud computing applications. *IEEE Access* 9:41731–41744. <https://doi.org/10.1109/ACCESS.2021.3065308>
11. Saleh H, Nashaat H, Saber W, Harb HM (2018) IPSO task scheduling algorithm for large scale data in cloud computing environment. *IEEE Access* 7:5412–5420. <https://doi.org/10.1109/ACCESS.2018.2890067>
12. Talha A, Malki MOC (2023) PPTS-PSO: a new hybrid scheduling algorithm for scientific workflow in cloud environment. *Multimed Tools Appl* 82(21):33015–33038
13. Mishra K, Majhi SK (2023) A novel improved hybrid optimization algorithm for efficient dynamic medical data scheduling in cloud-based systems for biomedical applications. *Multimed Tools Appl* 82(18):27087–27121
14. Kothi Laxman RR, Lathigara A, Aluvalu R, Viswanadhula UM (2022) PGWO-AVS-RDA: an intelligent optimization and clustering based load balancing model in cloud. *Concurr Computat: pract exper* 34(21):e7136
15. Daming L, Qinglang S, Lianbing D, Kaicheng C, Zhiming C, Mohammed BO (2020) Load balancing mechanism in the cloud environment using preference alignments and an optimisation algorithm. *IET Commun* 14(3):489–496
16. Jena U, Das P, Kabat M (2020) Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *J King Saud Univ-Comput Inf Sci* 34(6):2332–2342
17. Ebadifard F, Babamir SM (2018) A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurr Comput: pract exper* 30(12):e4368
18. Priya V, Kumar CS, Kannan R (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. *Appl Soft Comput* 76:416–424
19. Pourghaffari A, Barari M, Sedighian KS (2019) An efficient method for allocating resources in a cloud computing environment with a load balancing approach. *Concurr Comput: pract exper* 31(17):e5285
20. Zhou Z, Li F, Zhu H, Xie H, Abawajy JH, Chowdhury MU (2020) An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Comput Appl* 32(6):1531–1541
21. Haidri RA, Katti CP, Saxena PC (2019) Capacity based deadline aware dynamic load balancing (CPDALB) model in cloud computing environment. *Int J Comput Appl* 43(10):987–1001

22. JafarnejadGhomi E, Rahmani AM, Qader NN (2019) Service load balancing, scheduling, and logistics optimization in cloud manufacturing by using genetic algorithm. *Concurr Comput: pract exper* 31(20):e5329
23. Alla HB, Alla SB, Touhafi A, Ezzati A (2018) A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Clust Comput* 21(4):1797–1820
24. Devaraj AFS, Elhoseny M, Dhanasekaran S, Lydia EL, Shankar K (2020) Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *J Parallel Distrib Comput* 142:36–45
25. Mir Salim U, Islam AK, Yu-Chen Hu (2021) Context-Aware Scheduling in Fog Computing: a survey, taxonomy, challenges and future directions. *J Netw Comput Appl* 180:103008
26. Karunakaran V (2019) A stochastic development of cloud computing based task scheduling ALGORITHM. *J Soft Comput Paradigm (JSCP)* 1(01):41–48
27. Suresh A, Varatharajan R (2019) Competent resource provisioning and distribution techniques for cloud computing environment. *Clust Comput* 22(5):11039–11046
28. Ahmad MO, Khan RZ (2019) Pso-based task scheduling algorithm using adaptive load balancing approach for cloud computing environment. *Int J Sci Technol Res* 8(11):457–462
29. Lawanyashri M, Balusamy B, Subha S (2017) Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications. *Inform Med Unlocked* 8:42–50
30. Pourghaffari A, Barari M, Sedighian KS (2019) An efficient method for allocating resources in a cloud computing environment with a load balancing approach. *Concurr Comput: Pract Exper* 31(17):e5285
31. Gohil BN, Patel DR (2018) A hybrid GWO-PSO Algorithm for Load Balancing in Cloud Computing Environment. In 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT) (pp 185–191) IEEE
32. Kennedy J, Eberhart R (1995) Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks (Vol 4, pp 1942–1948) IEEE
33. Xingjun L, Zhiwei S, Hongping C, Mohammed BO (2020) A new fuzzy-based method for load balancing in the cloud-based Internet of things using a grey wolf optimization algorithm. *Int J Commun Syst* 33(8):e4370
34. Abualigah L, Diabat A (2020) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Clust Comput* 24:205–223
35. Bali MS, Alroobaea R, Algarni S, Alsafyani M, Mohiuddin K, Gupta K, Gupta D (2023) An efficient task allocation framework for scheduled data in edge based Internet of Things using hybrid optimization algorithm approach. *Phys Commun* 58(102047):102047. <https://doi.org/10.1016/j.phycom.2023.102047>
36. Chen X et al (2020) A WOA-Based Optimization Approach for Task Scheduling in Cloud Computing Systems. *IEEE Syst J* 14(3):3117–3128. <https://doi.org/10.1109/JSYST.2019.2960088>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.



Meena Malik received Ph.D. from Maharishi Dayanand University, Rohtak, Haryana, India in 2019. She is an Associate Professor in the department of Computer Science and Engineering at Chandigarh University, Mohali, Punjab, India. She has published over 20 international journals and conference papers in highly reputed journals. Her research interests include blockchain, cybersecurity, IoT security, and network and cloud security.



Durgesh Nandan a recipient of the prestigious "JSS Fellowship" from 2014 to 2018 and IEEE senior membership in 2020, is an Assistant Professor (SG) at Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune. He has a diverse professional background, working as a Guest faculty member at NIT Patna, an Account Manager in Accendere Knowledge Management Services Pvt. Ltd., and an AI consultant at deep Cognition. Dr. Nandan is an editorial board member for various SCI and Scopus journals and has over 100 research papers, 7 Indian patents, and 4 books to his credit. His research interests span computer arithmetic, VLSI architecture, speech processing, hardware architecture for real-time big data/AI applications, and Internet of Things (IoT).



Chander Prabha is currently working at Chitkara University Institute of Engineering and Technology, Chitkara University, Rajpura, Punjab, India. Her area of specialization includes Cloud Computing, Security, Opportunistic Networks, Data Science and Analytics, Fog Computing, and AIML. She has 20 Years of teaching and research experience. She can be contacted at prabhanice@gmail.com



Mueen Uddin (Senior Member, IEEE) received Ph.D. from UniversitiTeknologi Malaysia (UTM), in 2013. He is an Associate Professor in data and cybersecurity with the University of Doha for Science and Technology, Qatar. He has published over 150 international journals and conference papers in highly reputed journals with a cumulative impact factor of over 300. His research interests include blockchain, cybersecurity, IoT security, and network and cloud security.




Biswaranjan Acharya (Senior Member, IEEE) is working as an Assistant Professor at the Department of Computer Engineering-AI and Big Data Analytics. He received the M.C.A. degree from IGNOU, New Delhi, India, in 2009, and the M.Tech. degree in computer science and engineering from the Biju Pattanaik University of Technology (BPUT), Rourkela, Odisha, India, in 2012. He is currently pursuing a PhD. degree in computer application from the Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. He has more than ten years of experience in both academia at some reputed universities, such as Ravenshaw University and the software development field. He has published many research articles in international reputed journals and served as a reviewer for many peer-reviewed journals. He has more than 50 patents on his credit. His research interests include multiprocessor scheduling along with different fields, such as data analytics, computer vision, machine learning, and the IoT. He is also associated with various educational and research societies, such as IACSIT, CSI, IAENG, and ISC.



Yu-Chen Hu received his PhD. Degree in computer science and information engineering from the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan. Currently, Dr. Hu is a full professor in the Department of Computer Science at Tunghai University, Taiwan. He is a senior member of IEEE. He is also a member of Computer Vision, Graphics, and Image Processing (CVGIP), Chinese Cryptology and Information Security Association (CCISA), Computer Science and Information Management (CSIM) and Phi Tau Phi Society of the Republic of China. He joins the editorial boards of *Advances in Multimedia* (Hindawi), *Algorithms* (MDPI), *Electronics* (MDPI), *IET Image Processing*, *Intelligent Automation & Soft Computing* (Tech Science Press), *Mathematical Problems in Engineering* (Hindawi), etc. His research interests include data compression, image processing, information hiding, information security, computer network, deep learning, and bioinformatics.

Authors and Affiliations

Meena Malik¹ · Durgesh Nandan² · Chander Prabha³ · Mueen Uddin⁴ ·
Biswaranjan Acharya⁵ · Yu-Chen Hu⁶ 

✉ Yu-Chen Hu
ychu@thu.edu.tw; yuchen.martin.hu@gmail.com

Meena Malik
meenamlk@gmail.com

Durgesh Nandan
durgeshnandano51@gmail.com

Chander Prabha
prabhanice@gmail.com

Mueen Uddin
mueenmalik9516@gmail.com

Biswaranjan Acharya
biswaranjanachary2020@gmail.com

¹ Chandigarh University, Mohali, India

² School of CS & AI, SR University, Warangal 506371, India

³ Chitkara University Institute of Engineering and Technology, Chitakara University, Rajpura, India

⁴ College of Computing and IT, University of Doha for Science & Technology, Doha, Qatar

⁵ Department of Computer Engineering, AI & BDA, Marwadi University, Rajkot, Gujarat, India

⁶ Department of Computer Science, Tunghai University, No. 1727, Sec. 4, Taiwan Boulevard, Xitun District, Taichung City 407224, Taiwan R.O.C.