Wireless Personal Communications (2022) 125:1125–1144
https://doi.org/10.1007/s11277-022-09592-3

Lateral Wolf Based Particle Swarm Optimization (LW-PSO) for Load Balancing on Cloud Computing

Meena Malik[1] · **Suman**[1]

Accepted: 7 February 2022 / Published online: 15 March 2022

Abstract
Cloud services are rapidly evolving and has become a demand. Consequently, Load Balancing (LB) is needed to enhance the use of resources by optimal distribution of workload among various Virtual Machines (VMs). This study intends to solve the task scheduling issues and provide optimal LB to all the VMs by implementing the proposed hybrid Lateral Wolf and Particle Swarm Optimization (LW-PSO). The study aims to find the optimized VMs by the proposed hybrid methodology. It also intends to perform parallel task scheduling, thereby minimizing the response time and afford results quickly for each of the assigned tasks. The study uses Lateral Wolf (LW) to perform task scheduling in a parallel way and the Particle Swarm Optimization (PSO) obtains the optimal solution based on LW so as to find the optimized VMs. This creates flexibility among the VMs as they are neither overloaded nor under-loaded. All the VMs are equally assigned tasks. The proposed LW finds the Fitness Value (FV) and save this value. Then, it is fed to PSO and the best particle is updated along with its position and velocity. This process helps to find the optimized VMs and assign loads in accordance with the obtained optimal solution. The performance analysis is carried out by considering significant parameters such as average load, processor utilization, and average turnaround time, average response time, runtime and memory utilization. The analytical results show that the proposed method performs effectively than the existing system with respect to the mentioned parameters.

Keywords Cloud services · Load balancing · Task scheduling · Lateral wolf · Particle swarm optimization

1 Introduction

The cloud environments involve numerous requests that arise from varied geographical channels concurrently. These requests are randomly assigned to different cloud providers. This ensures the unequal load assignment at a specific node that is few nodes are overloaded. Whereas, few nodes are under-loaded. This unequal load might degrade the system performance. To solve this, correct Load Balancing (LB) is significant. The study [1]

✉ Meena Malik
meenamlkmlk@gmail.com

[1] Computer Science and Engineering, Chandigarh University, Mohali, India

reviewed various approaches by a comparative analysis of LB in the cloud. It affords an extensive opportunity for investigators to develop effective LB algorithms for the cloud environment [2]. The survey has been carried out by considering algorithms like heuristic-based LB (Round Robin algorithm, Improved Max–Min algorithm, Balanced Reduce algorithm and others.), metaheuristic based LB algorithm (Firefly algorithm, Honey Bee algorithm, Artificial Bee colony and others) and hybrid-based LB algorithms (Genetic Algorithm with Tabu search and so on). As LB is a strategy of detecting under-loaded and overloaded nodes, the article [3] explored a study that relies on the modern LB algorithms evolved particularly to fit the cloud environment. Taxonomy has been introduced for the LB algorithm. The simulation has been undertaken in the CloudSim simulator to assess the performance of the algorithms used in this study. These approaches should be clearly understood for future research. The proposed algorithms have to be implemented in real-time along with comparative evaluation [4]. Similarly, the paper [5] discussed various LB algorithms for cloud environments in a systematic way in accordance with a taxonomy. It has been found that irrespective of the significance of the network bandwidth in this environment, the high rate of network traffic led to degradation in the efficient usage of networking elements. This results in serious issues like data loss, communication delay and network failure. Thus, many effective LB algorithms have to be developed to use the network bandwidth in an effective way [6]. Additionally, the paper [7] introduced a hybrid LB strategy named Throttled and Equally Spread Current Execution. This approach has been utilized to reduce the time for response, enhance the services, minimize the cost of the machine and avoid bottleneck issue. In the same way, an improved scheduling algorithm has been proposed in this article [8]. This algorithm relies on the Particle Swarm Optimisation (PSO) algorithm. It intended to schedule huge tasks without affecting the performance of the system. Effective outcomes have been attained. Yet, the proposed methodology failed to consider certain objective functions like energy consumptions or cost. Moreover, this algorithm has to be altered to work as a concurrent workflow scheduling [9]. Thus, the present study intends to optimize the Virtual Machines (VM) by hybrid Lateral Wolf and Particle Swarm Optimization so as to avoid overloading and under-loading of VMs, thereby performing concurrent scheduling.

The major contributions of this study are listed below.

- To determine the optimized Virtual Machines by the proposed hybrid Lateral Wolf and Particle Swarm Optimization (LW-PSO) for efficient Load Balancing.
- To minimize the response time of the task and perform parallel scheduling by the proposed hybrid algorithm.
- To analyze the system performance with respect to various parameters (average load, average response time, average turnaround time, CPU utilization, runtime and memory utilization) to evaluate its efficiency.

1.1 Paper Organisation

The paper is organized as follows.

Section I discusses the basic concepts of Virtual Machine, Task scheduling and Load Balancing in cloud computing environment. This is followed by section II that discusses the review of the various existing systems for LB in cloud. Then, section III explores the proposed hybrid method to accomplish concurrent LB. The results obtained through the

proposed method is discussed in section IV. Finally, the overall study is summarized in section V.

2 Review of existing work

Various methodologies used by the existing system for Load Balancing (LB) in cloud computing are discussed in this section.

Task scheduling in an effective way is a major issue in the cloud computing environment. The study [10] has proposed a method to dynamically balance the load in a cloud environment via a combination of Modified Particle Swarm Optimization (MPSO) and an improved Q-learning algorithm named QMPSO. Actually, the combination process was done by adjusting the velocity of MPSO via gbest and pbest depending on the actions of Q-learning. This algorithm results in an improvement in makespan, throughput and energy utilization. On the other hand, LB also reduces waiting time for the tasks comparatively when executed separately as per this study. Future work is proposed to carry out load balancing among other tasks in a dynamic way. Moreover, the primary tasks in a virtual system are to schedule tasks effectively. The upcoming study aids with a proposed method for static task scheduling depending upon particle swarm optimization (PSO). Here, the tasks are independent and cannot be assigned properly. Thus, the LB technique is used to improve the performance of the PSO method. The results show improvement in resource utilization by 22% and a decrease in the makespan by 33%. The future work intends to consider cost reduction and fault tolerance capability [11]. Hence, it is clear that the Cloud environment is using task scheduling and load balancing for optimal file sharing. The article [12] has proposed a fuzzy-based multi-dimensional resource scheduling model in order to sustain the efficiency of resource scheduling in a cloud environment. The model was tested via simulations created by cloud simulators and showed effective results of 7% for an increase in resource scheduling and a 35% increase in the time factor. The future work was to propose a resource scheduling in privacy-aware resources with load balancing by considering success rate and response time [13].

Now-a-days, green cloud computing is facing certain issues like high energy cost and low efficiency. To avoid those issues, the following study [14] has given a global optimization algorithm known as resource-aware load balancing clonal algorithm focusing on task scheduling in order to deal with energy consumption in green cloud computing. The results prove that this algorithm performs effectively and exploration and exploitation factors were enhanced and balanced well. The future work is to have different energy-efficient approaches applicable for large data centres. Cloud computing is concentrating recently on the allocation of resource management efficiently. The paper [15] has introduced a method of combining fuzzy logic, evolutionary algorithms and task scheduling techniques in order to improve resource allocation in a cloud environment along with maintaining LB [16]. This method was quite effective in response time, task execution time and less energy consumption. Future works depend upon creating evolutionary algorithms in task schedulers and using bounding techniques for LB maintenance. Similarly, the article [17] has proposed PSO based scheduling algorithm via load balancing. Here the tasks are very diverse in the area. The proposed method has proven to be effective when compared to traditional methods. Yet, the study lacks real-time implementation. Cloud technology is currently facing optimization problems along with LB. In order to resolve this issue, this study [18] has proposed a Modified Adaptive Neuro-Fuzzy Inference System (MANFIS) for handling

dynamic LB. MANFIS is introduced by using the Fire-Fly algorithm. The results are up to the mark and satisfying by utilizing resources properly. Reinforcing the same concept is yet to be done. Additionally, the paper [19] has provided an approach known as the hybrid fruit fly optimization technique depending on the improvement of optimal resource utilization and reduced energy consumption along with reduced cost in the cloud. The experiment has resulted in efficient outcomes compared to existing LB algorithms. Future work was proposed to work with the QoS factor and balancing workflows in cloud computing.

Now-a-days, Mobile Cloud Computing (MCC) has become a trend, but these are also facing LB issues. As a way to avoid those issues, the following study [20] has proposed a strategy of Krill Load Balancer (Krill LB). They concentrate on VMs by increasing the throughput of the networks up to the level. This algorithm is depending upon factors like speed, task, cost and weight. This method has proved to work effectively against Honey Bee Behaviour Load Balancing (HBB-LB), Kill herd, Round Robin algorithms. Moreover, cloud computing is considered to be flexible and scalable and available at low cost. However, major issues come across these environments like LB and task scheduling. Thus, the study [21] is proposing a novel algorithm MGGS (modified genetic algorithm (GA) combined with greedy strategy). This method is used for task scheduling and uses a fewer number of iterations. The results showed that MGGS performs well compared to others. The future work is to implement a stochastic service system to improve the cloud resource environment. Correspondingly, the paper [22] has proposed to concentrate on CPU bound requests so as to avoid LB. The requests are divided into CPU-bound and I/O bound request. This is attained by using the CloudSim tool. The results are much effective in comparison to round-robin, honey bee, Naïve Bayesian approaches. Implementing this approach in geographical clouds having dispersed data centres is proposed in future work. The allocation of cloud resources accordingly is a major constraint in the cloud system. In order to support the above-said issue, the article [23] concentrated on LB and QoS methods performed in the cloud environment. The work observed Qos management applications along with their performance challenges and different models available using different tool kits available. The end results were effective. There are scopes for research opportunities in the area of LB in the cloud environment.

Another type of approach for handling load balancing is suggested in the article [24]. They are Capacity Based Deadline LB algorithm which is proposed to overcome the problem of LB. This algorithm ensures that customer needs are satisfied with minimal cost. The results are stated that this algorithm overcomes all hurdles and are effective than other competitive algorithms. Yet, it did not consider the QoS parameters and workflow applications for LB. Since cloud computing has grown out largely, the LB issue can be overlooked by larger approaches. Hence, the study [25] has proposed an algorithm named Region Rerouting Load balancing (RRRL) for dealing with the loading capacity issues. This approach achieved its aim with minimal latency and high throughput. The future work is to establish this algorithm in cloud servers across geographical regions. Cloud manufacturing has caught more attention recently. To make it successful, issues like scheduling and service selection (SOSL) has to be taken care of. To support this, the paper [26] has proposed a genetic algorithm implementing both Cloud Manufacturing Queuing Systems (CMfgQS) and the LB heuristic algorithm based on task process time (LBPT). This code is generally applied to larger data resulting in an effective outcome. Several more heuristics and Metaheuristics algorithms have to be compared accordingly in the near future. Likewise, the study [27] has proposed two hybrid metaheuristic algorithms. It used fuzzy logic with the PSO algorithm (TSDQ-FLPSO). Subsequently, it used simulated annealing with the PSO algorithm (TSDQ-SAPSO). The proposed algorithm has provided maximum results

for high dimensional problems along with great advantage of time, queue length, cost and resource utilization. LB is focusing on managing workloads by distributing resources among computers. So as to achieve that, the paper [28] introduced a combination of Firefly and Improved Multi Objective PSO technique abbreviated as (FIMPSO). The former is used to perform Search Space (SS) minimization and latter is used to find the improvised response. The results shows that hybrid algorithms results with maximum utilization of CPU up to 98%, memory utilisation (93%), 67% of reliability and throughput of 72%. This has improved the results comparing to other methods. Parallel method are suggested to overcome the issue of LB. One such approach has been proposed based on PSO [29]. The results of this paper gives a strong optimal results than others methods. The outcomes are simple, easily accessible and gives better solutions. LB and scheduling based applications setbacks are approached in various ways. Among them, the following approach as proposed in the paper [30] employed a hybrid algorithm based on gravitation search and non-dominated sorting genetic algorithm (GSA and NGSA). The results are proved effective in proposed algorithm's efficiency. Future work was to solve power consumption in cloud data centres and implementing green cloud computing. Resources can also be allocated using the study discussed in [31] as it has proposed allocation of resource using modified cloud resource provisioning algorithm aided by optimization technique and OCRP algorithm. The result of this paper reduced the optimal cost. In addition, it used low memory usage compared to other techniques. Yet, this study have to consider various parameters to confirm its efficacy further.

3 Proposed Methodology

The study proposed a hybrid Lateral Wolf and Particle Swarm Optimization (LW-PSO) to find the optimized VM so that tasks can be assigned to all the VMs by avoiding overloading and under-loading. The task scheduling and LB are performed in a cloud environment by implementing the proposed hybrid algorithms. This is carried out by various processes and is given in the below Fig. 1.

At first, the user request is taken as a task queue on VM and provided to the VM Manager. Then, the VMs are created. Subsequently, the active VMs are uploaded with tasks. By using the PSO optimization, the parameters are initialized. This is followed by initializing the particle with position and velocity to solve the target problem. After this, the fitness value is updated by LW operation. In LW, the starting population is initialized and the fitness is evaluated. The search agent is then updated in a lateral manner to find the fitness value of the search agents. Finally, the best fitness value is saved. The best particle,

position and velocity is updated to obtain the optimal solution. Thus, the high load VMs are optimized using the proposed hybrid LW-PSO by assigning tasks to low loaded VMs, thereby performing LB. The proposed system is also analyzed with respect to various performance measures, namely runtime, average load, CPU utilization, average turnaround time, average response time and memory utilization to evaluate its effectiveness.

3.1 Particle Swarm Optimization (PSO)

PSO is an optimization and meta-heuristic technique employed to optimize an issue by iteratively attempting to enhance a candidate resolution in accordance with the given quality measure. It rectifies an issue by possessing a candidate solution populations (particles). Then, it

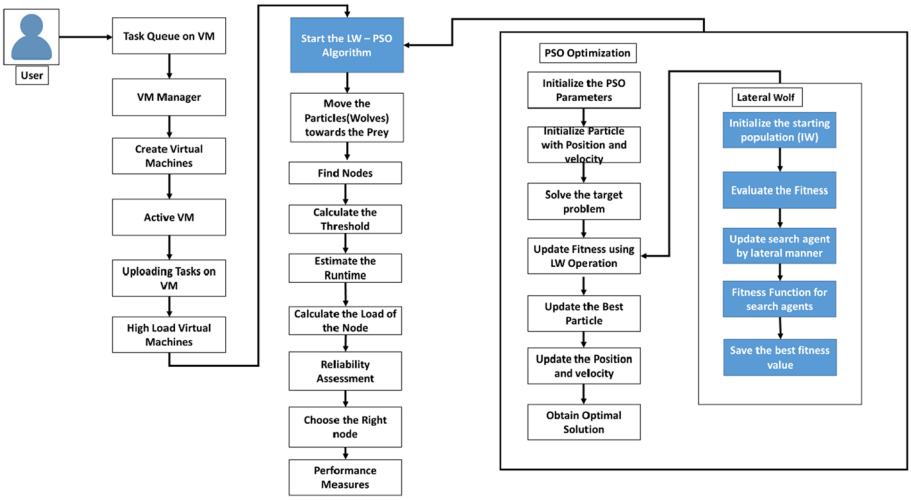

Fig. 1 Overall flow: lateral Wolf Based PSO Optimization for Load Balancing on Cloud Computing

moves these particles round the Search Space (SS) based on precise mathematical formula over the position as well as the velocity of the particle. The movement of an individual particle is inclined by its local best-identified position. On the other hand, it is directed towards the best-identified positions in SS. This is then updated as optimal positions and is expected for moving the swarm to optimal solutions. This particular algorithm does not utilize the gradient corresponding to the issue that is being optimized. This means that it does not need the optimization issue to be diverse as needed by classific optimization techniques like quasi newton and gradient descent methodologies. This study uses PSO to obtain the optimal solution for LB and task scheduling which are the main issues in the cloud environment.

At first, the position and velocity of the particle is initialized as per Eqs. 1 and 2.

$$\text{Position} : X_i = \left(x_{i,1}, x_{i,2}, \ldots, x_{i,n}\right) \varepsilon R^n \tag{1}$$

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

Subsequently, the velocity of the particle is computed by Eqs. 3 and 4.

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

$$V_i(k+1) = \omega \times V_i(k) + c_1 \times \text{random}_1 \times \left(\text{PBest}_i - x_i(k)\right) + c_2 \times \text{random}_2 \times \left(\text{GBest}_i - x_i(k)\right) \tag{4}$$

Here, w, c_1, c_2 are the constants, $\text{random}()_1, \text{random}()_2$ indicates the random variable. After updation, the position and velocity of the particle is given by Eqs. 5 and 6.

$$X_i(k+1) = X_i(k) + V_i(k+1) \tag{5}$$

$$X_i(k+1) = \text{Inertia} + \text{cognitive} + \text{social} \tag{6}$$

The pseudo-code corresponding to PSO algorithm is given below.

Algorithm I-PSO
Step 1: For individual particle Initialize the particle End Do **Step 2:** For individual particle Compute the fitness value (FV) **Step 3:** If FV is optimal than the best FV (p1Best) in history Set present FV as new p1Best End **Step 4:** Select the particles with optimal FV of every particle as g1Best **Step 5:** For individual particle Compute Particle Velocity (PV) Update PV End While minimum error or maximum iterations is not obtained

All the particles are initialized. The fitness value (FV) for each of the particles is computed. When the computed FV is better when compared to the optimal FV in history, this particular FV is set as a new FV. Finally, all the particles that possess the best FV are selected. The velocity of these particles are computed and later updated. This process continues until maximum iterations, or minimum error is obtained.

3.2 Lateral Wolf (LW)

The LW algorithm imitates the hunting mechanism and leadership hierarchy of a grey wolf. Four kinds of grey wolves, namely alpha, omega, beta and delta are applied to simulate this leadership hierarchy. Additionally, three main hunting steps are implemented to accomplish optimization. These steps include searching the prey, encircling and attacking it. The proposed LW has recently evolved as an effective meta-heuristic optimization method. It has the ability to solve unimodal issues, works effective for complex functions, thereby eluding local minima. It also has a superior exploration capacity for multi-modal issues.

In addition, the encircling process of the mathematical model is given by Eqs. 7 and 8.

$$d = \left| cx_p(t) - x(t) \right| \tag{7}$$

$$x(t+1) = x_p(t) - ad \tag{8}$$

Here t denotes the current iteration count, aand c denotes the coefficient vectors, x_p is the prey position vector and x denotes the LW's position vector. Here the aand c vectors are computed by Eqs. 9 and 10.

$$a = 2.f.r_1 - a \tag{9}$$

$$c = 2.r_2 \tag{10}$$

Here f is linearly minimized from two to zero via iteration. The random vectors are given by r_1 and r_2 within the range of (0, 1).

To perform mathematical simulation of the LW behaviour, an assumption is made that the beta, delta and alpha (availability of best solution) are effectively informed regarding the potential prey's location. The three of the effective solutions attained is stored, other agents are enforced to update their corresponding positions in accordance with the locations of the best agent. This association is given by the below equations.

$$d_\alpha = \left|c_1 x_\alpha(t) - x\right|, d_\beta = \left|c_2 x_\beta(t) - x\right|, d_\delta = \left|c_3 x_\delta(t) - x\right| \tag{11}$$

$$x_1 = x_\alpha - a_1(d_\alpha), x_2 = x_\beta - a_2(d_\beta), x_3 = x_\delta - a_3(d_\delta) \tag{12}$$

$$x(t+1) = \frac{x_1 + x_2 + x_3}{3} \tag{13}$$

The pseudo-code corresponding to LW algorithm is given below.

Algorithm II-LW

Input: Size of the problem, Size of the population
Output: Pg1_best
Step 1: Start
Initialize the grey wolves population Y_j where (j=1, 2, …., n)
Initialize b, B and C
Compute the FV of agent grading and search agents
Y_α is the **optimal** solution in a search agent, Y_β is the next optimal solution in a search agent and Y_δ is the third optimal solution in a search agent.
Step 2: Set $t1 = 0$
While (t1 < maximum iteration count)
For individual search agent
Update the current position of the search agent
End for
Step 3: Update b, B and C in a lateral manner
Compute the FV of each search agents, grading
Update Y_α, Y_β and Y_δ
$t1 = t1 + 1$
End while
End

The size of the problem and the size of the population is taken as input. At first, the grey wolf's population are initialized. Then, the FV of agent grading and search agents are computed. Subsequently, the present position of the search agent is updated when t1 is less than the maximum iteration count as per step 2. Finally, the best FV Pg1_best is updated.

3.3 Lateral Wolf Based Particle Swarm Optimization (LW-PSO)

In this study, the Particle Swarm Optimization and Lateral Wolf are hybrid for LB in the cloud environment. The PSO has various merits like easy implementation, simple concept, computational efficiency and robustness in terms of parameter control. Due to these merits, it is efficient than other methods. Thus, this study employed this particular algorithm to determine the optimized VM so as to schedule tasks, thereby minimizing overloading and under-loading accordingly. On the other hand, a solution is required to minimize the probability of PSO to trap into Local Minimum (LM). In this study, LW is employed to assist the PSO algorithm in minimizing this LM issue, thereby performing efficiently in combination with PSO to schedule the tasks in the cloud environment. The LW algorithm has the ability to avoid these threats by directing few particles to particular positions (partially enhanced by the LW algorithm) without directing them to some random position. Additionally, the LW algorithm has the capacity to perform task scheduling and LB in parallel, it needs only minimum memory, easy implementation and faster convergence which shows its effectiveness for implementation in this study.

The fitness value is computed by hybrid LW-PSO through the computations shown in Eqs. 14 and 15.

$$\text{In the Position Update } X_i(k+1) = X_i(k) + V_i(k+1) + d_\alpha = \left|c_n x_w(t) - x\right| \tag{14}$$

Here w − wolves, n − number.

$$\text{In the Position Velocity } X_i(k+1) = \text{Inertia} + \text{cognitive} + \text{socialx}(t+1) + \frac{x_1 + x_2 + x_3 \cdots + x_n}{n} \tag{15}$$

The pseudo-code of the proposed LW-PSO is given below.

Algorithm III-LW-PSO

Max_iter: the maximum iteration count fixed by the user
Pop_Size: the size of population count fixed by the user
$poss$: the minimum rate of possibility fixed by the user
Methodology LW-PSO
Step 1: Initialize the particles
for $i = 1\ to\ Max\_iter$ do
for $j = 1\ to\ Pop\_Size$ do
Run PSO
Update the position and the velocity of the present particle
Step 2: if $rand(0,1) < poss$ then // avoid local minima
Set b, B and C values
for $g = 1\ to\ 10$ do // minimum iteration count
for $n = 1\ to\ 10$ do // minimum population size
Run LW
Step 3: Update the α, β, γ wolves' positions in a lateral way
Update b, B and C values
End for
End for
Step 4: Position of present particle = mean of three optimal wolves' positions
End if
End for
End for
End methodology

The particles are initialized and iteration is performed by considering the maximum iteration count and the size of population count fixed by the user. Subsequently, the position and velocity of the present particle are updated. The LW algorithm is hybrid with PSO to avoid local minima and step 2 is implemented. Then, the positions of wolves are updated concurrently. This is continued until the position of the present particle is similar to the mean of the three optimal wolves' position.

4 Results and Discussion

The proposed LW-PSO is hybrid to perform Load Balancing (LB) in an efficient way. The obtained results from the implementation of this methods are discussed in this section.

Table 1 Analysis of the proposed and existing system [32] with respect to runtime

Run time(s)						
Task	100	150	200	250	300	400
EBCA-LB	77.9	98.55	61.75	165.5	192.25	221
HBB-LB	28.5	61.5	72.5	85.5	95	110
Existing	23.5	51	61.75	76.9	86.25	96.5
Proposed	20.25	46.89	57.16	70.65	76.39	89.57

4.1 Experimental Setup

The proposed system utilized a simulation environment named CloudSim using python language. This simulation tool permits the ability to explore the workflows as well as hosts.

4.2 Performance and Comparative analysis

The performance of this method is also analysed by comparing the proposed method with the existing methods with respect to average response time, memory utilization, average load, CPU utilization, average turnaround time and runtime.

The performance of the proposed system is assessed in terms of runtime by comparing it with Enhanced Bee Colony Algorithm (EBCA-LB) and existing methods. The obtained results are shown in the below Table 1.

The VM speed and task length are the two significant parameters to assign tasks to all the resources. Here various tasks are assigned. When 100 tasks are assigned, the Hybrid Enhanced Bee Colony Algorithm (EBCA-LB) takes 77.9 s, Hybrid Honey Bee Behaviour Load Balancing (HBB-LB) takes 28.5 s, and the existing method takes 23.5 s. Whereas the runtime of the proposed system is found to be 20.25 s when 100 tasks are assigned. Similarly, when various tasks are assigned, like 150 tasks, 200 tasks, 250 tasks, 300 and 400 tasks, the runtime of the proposed system is found to be low than the existing methods [32]. It is graphically shown in the below Fig. 2.

The execution time of the proposed and existing methods [32] for individual tasks are computed. It is found that the runtime of the proposed system is low when 100, 150, 200, 250, 300 and 400 tasks are assigned. This indicates that the runtime of the proposed system remains low than the existing system even when more tasks are assigned.

In addition, the proposed method is also analysed by comparing it with other existing methods like Hybrid Job Scheduling Algorithm (HJSA) and Task Scheduling Load Balancing Ant Colony Optimization (TSLBACO) in terms of runtime to evaluate its efficiency further. The results obtained are given in the below Table 2.

More tasks are assigned to VMs. The performance of the proposed and existing HJSA and TSLBACO [32] are analysed to find the efficient algorithm that determines the optimized VM and assigns tasks to them accordingly in minimum time. When 200 tasks are assigned, the runtime of the existing method is 0.5 s, the existing HJSA takes 0.61 s, and TSLBACO takes 0.63 s. On the other hand, the proposed system takes only 0.45 s. Likewise, the proposed system shows minimum runtime in comparison to other mentioned methods when 400, 600, 800 and 1000 tasks are assigned. Typically, as the tasks increase, the running time is more. But, here, the proposed method takes only minimum time for execution even when huge tasks are assigned. It is graphically shown in the below Fig. 3.

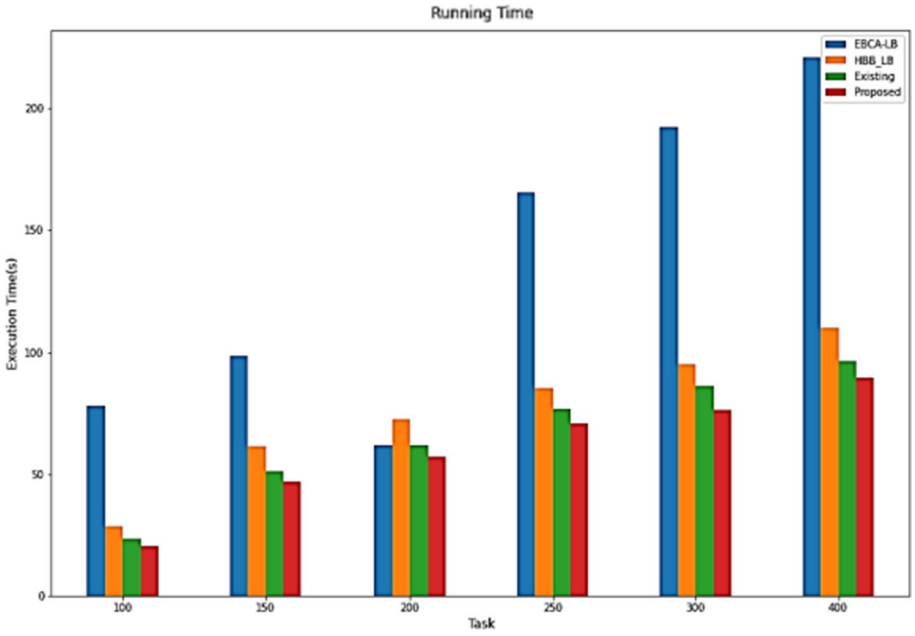

Fig. 2 The runtime of the existing [32] and the proposed method

Table 2 Analysis of the proposed and existing methods [32] with respect to runtime

Run time(s)					
Task	200	400	600	800	1000
Existing	0.5	0.58	0.63	0.7	0.75
HJSA	0.61	0.61	0.7	0.79	0.85
TSLBACO	0.63	0.71	0.75	0.83	0.95
Proposed	0.45	0.48	0.56	0.63	0.69

The proposed and existing HJSA, TSLBACO are assessed by assigning more tasks to VMs. The ability of each of these algorithms to find the optimized VM is examined when tasks are increased. The analytical results reveal that the proposed method takes minimum runtime for execution than existing methods. Due to the computational efficiency of PSO and parallel LB capacity of LW, the proposed hybrid LW-PSO accomplished minimum runtime proving its efficacy.

The proposed method is also assessed with respect to average load, response time, average turnaround time, CPU utilization and memory utilization. The obtained results are shown in the below Tables 3, 4 and 5.

It is found that the existing RR method shows an average load of 0.43 s, an average turnaround time of 41.98 ms and 30.5 s as average response time. Similarly, the existing Round Robin (RR), First Come First Service (FCFS), Short jobs First (SJF), Improved PSO (IPSO), Firefly IPSO (FF-IPSO) and Firefly Improved Multi-objective PSO (FIMPSO) shows high average load, response time and turnaround time. It is graphically shown in the below Figs. 4, 5 and 6.

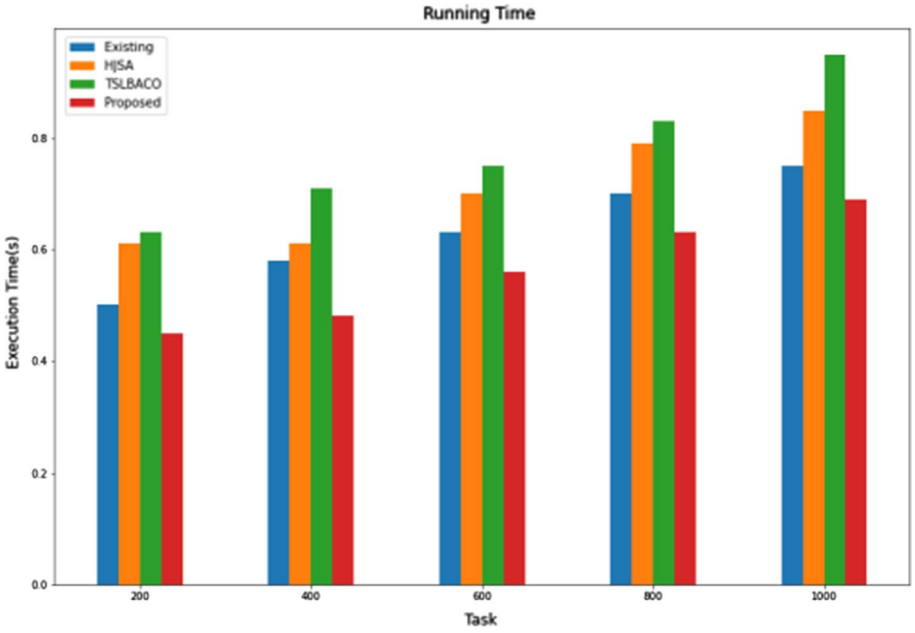

Fig. 3 Comparison of the proposed and existing methods [32] with respect to runtime

Table 3 Analysis of the proposed and existing methods [28] with respect to average response time

Techniques	Average load (ms)	Average turnaround time (ms)	Average response time (ms)
RR	0.43	41.98	30.5
SJF	0.495	41.56	30.24
FCFS	0.46	41.87	30.84
Firefly	0.47	55.54	48.87
IPSO	0.457	57.74	49.23
FF-IPSO	0.259	22.13	15.21
FIMPSO	0.247	21.09	13.58
Proposed	0.231	20.56	12.96

From Fig. 4, it is clear that the proposed hybrid LW-PSO shows a minimum average load in comparison to the existing FCFS, RR, SJF, IPSO, Firefly, FF-IPSO and FIMPSO. Similarly, the average turnaround time of the proposed and existing methods are compared and it is graphically shown in Fig. 5.

From the Fig. 5, it is found that the proposed method shows minimum turnaround time in comparison to the existing RR, FCFS, SJF, IPSO, Firefly, FF-IPSO and FIMPSO. Additionally, the average response time is also computed for the proposed method and it is compared with the mentioned existing methods to find its efficacy. The response time is a significant factor in the cloud environment. The results corresponding to this are shown in Fig. 6.

Table 4 Analysis of the proposed and existing methods [28] with respect to CPU utilization

Methods	Task types			
	Small	Medium	Large	Extra large
RD	45	50	64	70
WRR	48	58	67	75
DLB	50	63	70	80
LB-BC	57	68	75	85
LB-RC	64	75	85	90
IPSO-firefly	67	77	89	98
FIMPSO	70	79	95	99
Proposed	74	82	98	101

Table 5 Analysis of the proposed and existing methods [28] with respect to memory utilization

Methods	Task types			
	Small	Medium	Large	Extra large
RD	40	48	59	70
WRR	44	53	63	75
DLB	48	58	66	77
LB-BC	54	62	70	80
LB-RC	56	66	75	83
IPSO-firefly	57	70	80	86
FIMPSO	60	72	83	89
Proposed	63	76	86	90

The proposed system takes less time to respond to each user request thereby proving the system efficiency. This response time is also minimum when compared to the traditional methods. This outstanding performance of the proposed method makes it effective. Generally, hybrid methods take more time. But, the proposed hybrid algorithms have more computational efficiency. This makes them work faster.

The proposed hybrid LW-PSO is assessed in terms of CPU and memory utilization for different task types. It is shown in the Table 4.

Various kinds of tasks like small, large, medium and extra-large are considered. The existing Random method- RD utilized CPU only to a minimum extent in all kinds of tasks. Similarly, the CPU utilization of all the other existing methods namely Weighted Round Robin (WRR), Diffusive Load Balancing (DLB), Load Balancing Bayes and Clustering (LB-BC), Load balancing Resource control (LB-RC), IPSO-Firefly and FIMPSO are analysed by comparing with the proposed method. It is found that the proposed method efficiently utilized CPU at a maximum rate of 74% for small tasks, 82% for medium tasks, 98% for large tasks and 101% for extra-large tasks. It is graphically shown in the below Fig. 7.

The above Fig. 7 shows that the proposed method performs effective CPU utilization in comparison to other existing methods. Usually, as the tasks increases the CPU cannot be efficiently used. But, when the proposed hybrid LW-PSO is implemented, it has the ability to effectively perform CPU utilization which makes it suitable for load balancing

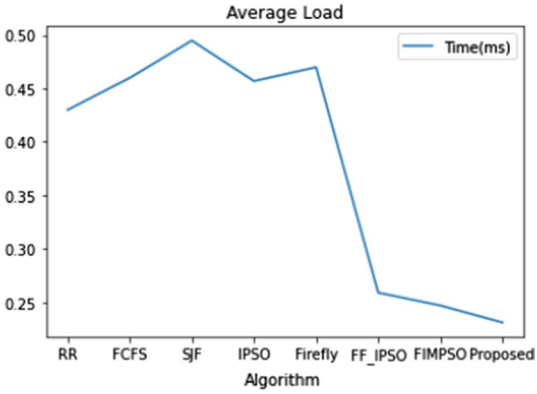

Fig. 4 Analysis of the proposed and existing methods [28] in terms of average load

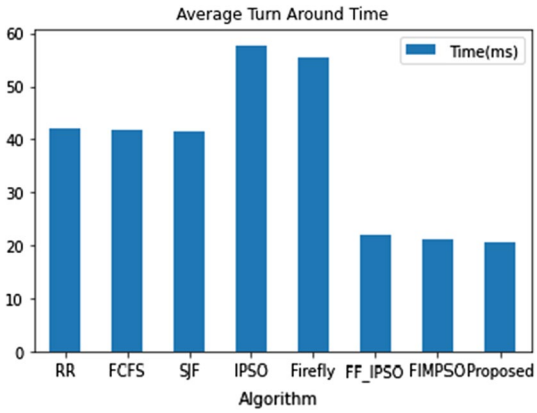

Fig. 5 Analysis of the proposed and existing methods [28] in terms of average turnaround time

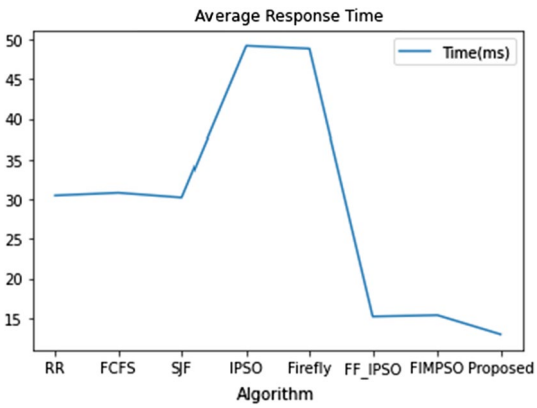

Fig. 6 Analysis of the proposed and existing methods [28] in terms of average response time

in cloud system. The proposed methodology is also analysed in terms of memory use as shown in Table 5.

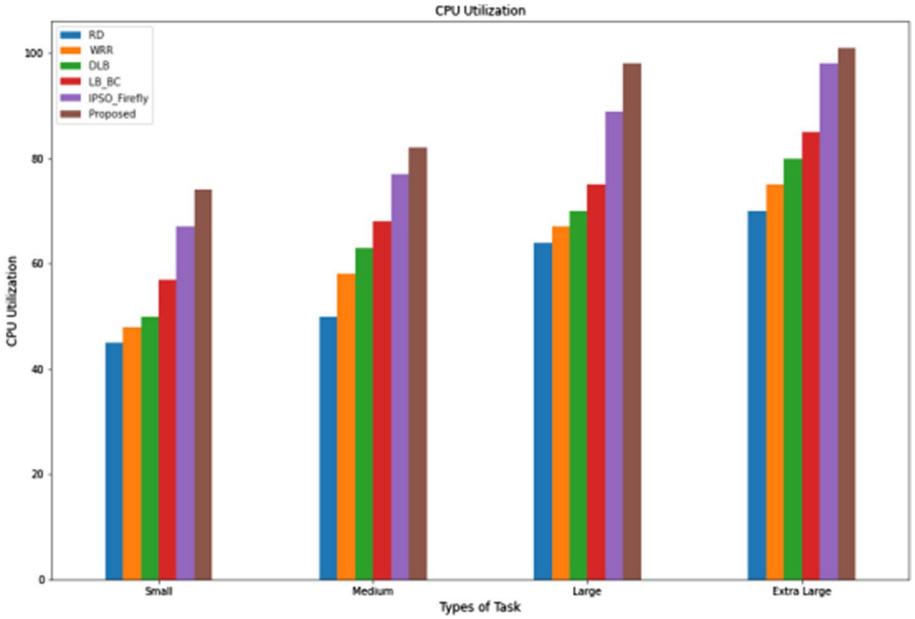

Fig. 7 Analysis of the proposed and traditional methods [28] with respect to CPU Utilization

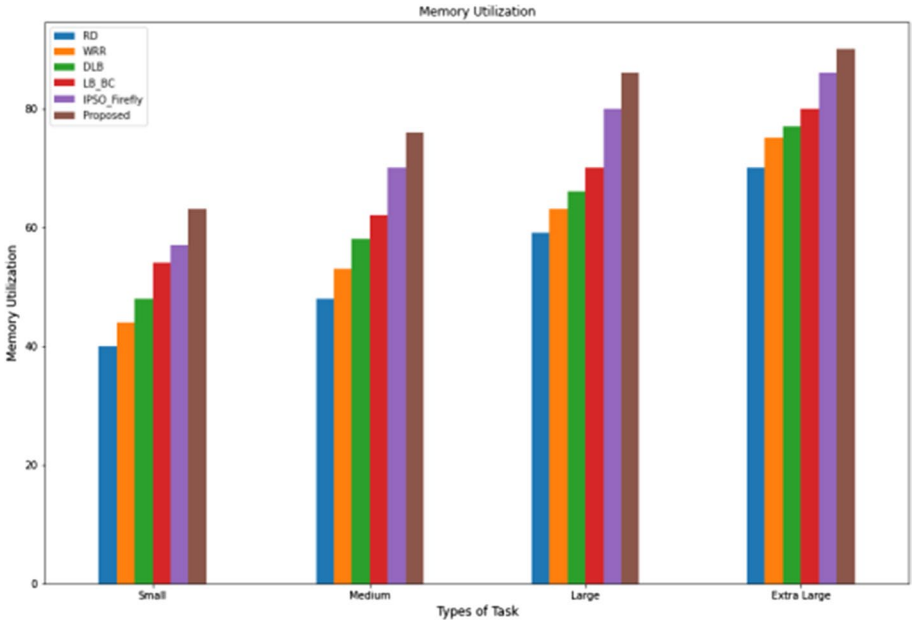

Fig. 8 Analysis of the proposed and traditional methods [28] with respect to memory Utilization

Similarly, the proposed system is compared with the traditional algorithms to find

Table 6 Parameter initialization

Parameters	Values
Max iteration	100
Higher and lower inertia weight	Higher = 0.5, Lower = 0.10
Training factors	3
Number of processors	2
Simulation parameters	
Input size (MI)	Depends on the task size
Output size (MI)	Depends on the Input size
Storage capacity	1,000,000
Bandwidth, Mbit/s	3000

its efficiency with regard to memory utilization. Various kinds of tasks as small, large, medium and extra-large are considered. The analytical outcomes explore that the proposed method uses 63% of memory for small tasks, 76% of memory for medium tasks, 86% of memory for large tasks and 90% of memory for extra-large tasks. This memory usage is efficient for proposed than existing methods as shown in Table 5. This means the proposed hybrid method efficiently uses memory for all the task types. It is graphically shown in the Fig. 8.

The proposed system shows efficient memory utilization in comparison to the conventional techniques. As the task increases, the efficiency of the memory utility remains steady for the proposed method than the existing method. This proves that the proposed method is consistent than the existing method even when more tasks are assigned for the VMs. It still has the ability to find the optimized VM and perform effective LB which response fast to the user request. Further, the parameter initialization for the proposed method are shown in Table 6.

The study [33] employed hybrid ant colony and lion optimization for solving the issues of task scheduling in cloud system. Though the study showed better outcomes it considered only minimum parameters such as imbalance degree, makespan time and response time. This minimum parameter is insufficient to prove its efficiency. The present study considered various significant parameters for analysis like runtime, average load, memory utilization, average response time, average turnaround time and CPU utilization. The analytical results are found to be efficient for the proposed system with regard to all the six significant parameters. This proves that the proposed hybrid LW-PSO is suitable to solve the tasks scheduling challenges in cloud environment.

5 Conclusion

In this study, the Lateral Wolf-Particle Swarm Optimization (LW-PSO) has been proposed for solving the task scheduling issues in cloud system by finding the optimized Virtual Machines (VMs). An analysis is performed by considering significant parameters like average load, CPU utilization, average turnaround time, runtime and average response time and memory utilization. The results explored that the proposed method is efficient than the existing methods as it reduces the average load, response time and turnaround time. The proposed method effectively used memory and CPU in comparison to the traditional methodologies. Though the tasks are increased and the types of tasks are complex, the

proposed hybrid LW-PSO accomplished effective results in minimum time by assigning tasks to VMs by avoiding overloading and under-loading. This performance is found to be outstanding than the conventional techniques. Thus, this study will assist to perform optimal Load Balancing (LB) of VMs thereby avoiding the task scheduling issues.

Acknowledgements None.

Authors contribution I Am MEENA MALIK Hereby State That The Manuscript Title Entitled "Lateral Wolf Based Particle Swarm Optimization (LW-PSO) For Load Balancing On Cloud Computing" Submitted To Wireless Personal Communications, I and my Co-author Suman Confirm That This Work Is Original And Has Not Been Published Elsewhere, Nor Is It Currently Under Consideration For Publication Elsewhere. And I Am Assistant Professor in Computer Science Department, Chandigarh University, Mohali.

Funding This research work was not funded by any organization/institute/agency.

Declarations

Conflict of interest I confirm that this work is original and has either not been published elsewhere, or is currently under consideration for publication elsewhere. None of the authors have any competing interests in the manuscript.

Consent to participate I confirm that any participants (or their guardians if unable to give informed consent, or next of kin, if deceased) who may be identifiable through the manuscript (such as a case report), have been given an opportunity to review the final manuscript and have provided written consent to publish.

References

1. Hota, A., Mohapatra, S., & Mohanty, S. (2019). Survey of different load balancing approach-based algorithms in cloud computing: A comprehensive review. *Computational Intelligence in Data Mining*. https://doi.org/10.1007/978-981-10-8055-5_10
2. Gabi, D., Ismail, A. S., Zainal, A., & Zakaria Z. (2017). Solving task scheduling problem in cloud computing environment using orthogonal taguchi-cat algorithm. *International Journal of Electrical & Computer Engineering (2088–8708)*, *7*(3).
3. Mishra, S. K., Sahoo, B., & Parida, P. P. (2020). Load balancing in cloud computing: A big picture. *Journal of King Saud University-Computer and Information Sciences, 32*(2), 149–158. https://doi.org/10.1016/j.jksuci.2018.01.003
4. Gamal, M., Rizk, R., Mahdi, H., & Elnaghi, B. E. (2019). Osmotic bio-inspired load balancing algorithm in cloud computing. *IEEE Access, 7*, 42735–42744. https://doi.org/10.1109/ACCESS.2019.2907615
5. Thakur, A., & Goraya, M. S. (2017). A taxonomic survey on load balancing in cloud. *Journal of Network and Computer Applications, 98*, 43–57. https://doi.org/10.1016/j.jnca.2017.08.020
6. Upadhyay, S. K., Bhattacharya, A., Arya, S., & Singh, T. (2018). Load optimization in cloud computing using clustering: A survey. *International Research Journal of Engineering and Technology, 5*(4), 2455–2459.
7. Subalakshmi, S., & Malarvizhi, N. (2017). Enhanced hybrid approach for load balancing algorithms in cloud computing. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 2*(2), 136–142.
8. Saleh, H., Nashaat, H., Saber, W., & Harb, H. M. (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
9. Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., & Alzain, M. A. (2021). A load balancing algorithm for the data centres to optimize cloud computing applications. *IEEE Access., 9*, 41731–41744. https://doi.org/10.1109/ACCESS.2021.3065308

10. Jena, U., Das, P., & Kabat, M. (2020). Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences*. https://doi.org/10.1016/j.jksuci.2020.01.012
11. Ebadifard, F., & Babamir, S. M. (2018). A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. *Concurrency and Computation: Practice and Experience., 30*(12), e4368. https://doi.org/10.1002/cpe.4368
12. Priya, V., Kumar, C. S., & Kannan, R. (2019). Resource scheduling algorithm with load balancing for cloud service provisioning. *Applied Soft Computing, 76*, 416–424. https://doi.org/10.1016/j.asoc.2018.12.021
13. Balaji, K., Kiran, P. S., & Kumar, M. S. (2021). An energy efficient load balancing on cloud computing using adaptive cat swarm optimization. *Materials Today: Proceedings*. https://doi.org/10.1016/j.matpr.2020.11.106
14. Lu, Y., & Sun, N. (2019). An effective task scheduling algorithm based on dynamic energy management and efficient resource utilization in green cloud computing environment. *Cluster Computing, 22*(1), 513–520.
15. Pourghaffari, A., Barari, M., & Sedighian, K. S. (2019). An efficient method for allocating resources in a cloud computing environment with a load balancing approach. *Concurrency and Computation: Practice and Experience, 31*(17), e5285. https://doi.org/10.1002/cpe.5285
16. Muthusamy, G., & Chandran, S. R. (2021). Cluster-based task scheduling using K-means clustering for load balancing in cloud datacenters. *Journal of Internet Technology, 22*(1), 121–130.
17. Ahmad, M. O., & Khan, R. Z. (2019). Pso-based task scheduling algorithm using adaptive load balancing approach for cloud computing environment. *International Journal of Scientific & Technology Research*, *8*(11).
18. Devi, T. D., Subramani, A., & Anitha, P. (2021). Modified adaptive neuro fuzzy inference system based load balancing for virtual machine with security in cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing, 12*(3), 3869–3876.
19. Lawanyashri, M., Balusamy, B., & Subha, S. (2017). Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications. *Informatics in Medicine Unlocked., 8*, 42–50. https://doi.org/10.1016/j.imu.2017.02.005
20. Hasan, R. A., & Mohammed, M. N. (2017). A krill herd behaviour inspired load balancing of tasks in cloud computing. *Studies in Informatics and Control, 26*(4), 413–424. https://doi.org/10.24846/v26i4y201705
21. Zhou, Z., Li, F., Zhu, H., Xie, H., Abawajy, J. H., & Chowdhury, M. U. (2020). An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. *Neural Computing and Applications., 32*(6), 1531–1541. https://doi.org/10.1007/s00521-019-04119-7
22. Ebadifard, F., & Babamir, S. M. (2020). Autonomic task scheduling algorithm for dynamic workloads through a load balancing technique for the cloud-computing environment. *Cluster Computing*. https://doi.org/10.1007/s10586-020-03177-0
23. Prakash, S. (2018). A literature review of QoS with load balancing in cloud computing environment. *Big Data Analytics* 667–75.
24. Haidri, R. A., Katti, C. P., & Saxena, P. C. (2019). Capacity based deadline aware dynamic load balancing (CPDALB) model in cloud computing environment. *International Journal of Computers and Applications*. https://doi.org/10.1080/1206212x.2019.1640932
25. Sekaran, K., & Krishna, P. V. (2017). Cross region load balancing of tasks using region-based rerouting of loads in cloud computing environment. *International Journal of Advanced Intelligence Paradigms, 9*(5–6), 589–603. https://doi.org/10.1504/ijaip.2017.088151
26. Jafarnejad Ghomi, E., Rahmani, A. M., & Qader, N. N. (2019). Service load balancing, scheduling, and logistics optimization in cloud manufacturing by using genetic algorithm. *Concurrency and Computation: Practice and Experience, 31*(20), e5329. https://doi.org/10.1002/cpe.5329
27. Alla, H. B., Alla, S. B., Touhafi, A., & Ezzati, A. (2018). A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Cluster Computing, 21*(4), 1797–1820.
28. Devaraj, A. F. S., Elhoseny, M., Dhanasekaran, S., Lydia, E. L., & Shankar, K. (2020). Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments. *Journal of Parallel and Distributed Computing., 142*, 36–45. https://doi.org/10.1016/j.jpdc.2020.03.022
29. Pradhan, A., Bisoy, S. K., & Das, A. (2021). A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. *Journal of King Saud University-Computer and Information Sciences.* https://doi.org/10.1016/j.jksuci.2021.01.003

30. Karunakaran, V. (2019). A stochastic development of cloud computing based task scheduling ALGORITHM. *Journal of Soft Computing Paradigm (JSCP), 1*(01), 41–48. https://doi.org/10.36548/jscp.2019.1.005
31. Suresh, A., & Varatharajan, R. (2019). Competent resource provisioning and distribution techniques for cloud computing environment. *Cluster Computing, 22*(5), 11039–11046. https://doi.org/10.1007/s10586-017-1293-6
32. Xingjun, L., Zhiwei, S., Hongping, C., & Mohammed, B. O. (2020). A new fuzzy-based method for load balancing in the cloud-based Internet of things using a grey wolf optimization algorithm. *International Journal of Communication Systems, 33*(8), e4370. https://doi.org/10.1002/dac.4370
33. Abualigah L, Diabat A (2020) A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments. *Cluster Computing* 1–19. https://doi.org/10.1007/s10586-020-03075-5

Dr. Meena Malik is Assistant Professor in the Department of Computer Science and Engineering of Chandigarh University, Punjab. She has completed PhD (CSE) from Maharshi Dayanand university, Rohtak, M. Tech(CSE) from Banasthali University, Rajasthan and B.Tech (CSE) from Kurukshetra University. She has qualified GATE 2012 and have four years of teaching experience. She has more than 10 research publications in international journals and conferences. She has collaborated actively with various FDPs seminars and national/international conferences organized by different academics institutions. Her research interests lie in the area of Computer Networks, Energy Efficiency in Wireless Sensor Networks and Security. Her goal is to attain excellence in academics and administration with positive attitude.

Suman Academic Profile: Masters: Department of Computer Science & Engineering, Kurukshetra University, Haryana. Graduation: NC College of Engineering, Panipat, Haryana. I am Suman is a Research Scholar from the Department of Computer Science and Engineering working in Chandigarh University, Punjab, India. My major area of research includes load balancing in cloud computing.