

A particle swarm optimization inspired global and local stability driven predictive load balancing strategy

Niladri Sekhar Dey, Hrushu Kesava Raju Sangaraju

Department of Computer Science and Engineering, KLEF, Vijayawada, India

Article Info

Article history:

Received Feb 25, 2024

Revised Apr 18, 2024

Accepted May 7, 2024

Keywords:

Cloud-based load balancing
Particle swarm optimization
Resource utilization analysis
Service level agreements
Space-time swarm positioning
Time complexity reduction

ABSTRACT

In distributed systems and parallel computing, optimal load balancing is difficult. These abstract addresses load balancing in distributed situations, highlighting current solutions' flaws and emphasizing the need for new ones. Load balancing research includes centralized and distributed algorithms, heuristics, and predictive models. Despite various successful methods, workload adaptability, overhead reduction, and scaling to large systems remain unresolved. This study proposes a particle swarm optimization (PSO) load balancing method that considers global and local stability considerations. The proposed method uses PSO principles to balance exploration and exploitation and allocate resources among distributed nodes. Predictive components improve preventative load management by predicting workload changes. Global and local load balancing stability criteria distinguish this study. The recommended method considers global system-wide performance indicators, local node-level characteristics, and micro-level stability to maximize system efficiency. A dual-focus technique distinguishes the proposed load balancing strategy from others, solving dynamic distributed system challenges. The study examines load balancing system advances and suggests improvements and further research. More accurate prediction modeling, stability measures, and application-specific enhancements may be studied in the future. Experimental validation and real-world implementation of the recommended approach are necessary to determine its practicality and ability to handle modern distributed computing systems.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Niladri Sekhar Dey

Department of Computer Science and Engineering, KLEF

Vijayawada, Andhra Pradesh, India

Email: sd.niladri@gmail.com

1. INTRODUCTION

In the dynamic field of modern computer systems, the main challenge is to achieve optimal resource use while maintaining system stability and performance. To address the challenges of task distribution and resource management in extensive distributed computing systems, innovative strategies must be devised. Load balancing is essential in this scenario as it ensures the equitable distribution of resources across the system to prevent congestion, decrease response times, and enhance overall efficiency. Predictive load balancing methods have transformed systems by allowing them to predict and preemptively handle fluctuations in demand, thus avoiding performance degradation or system failures. An new strategy in load balancing research and implementation involves integrating particle swarm optimization (PSO) ideas with global and local stability considerations.

This publication signifies the completion of this revolutionary technique. The framework tries to efficiently handle the intricate relationships among system dynamics, workload fluctuations, and resource limitations. The method employs the concepts of PSO, a bio-inspired optimization technique that relies on the collective behavior of swarming organisms. The system uses virtual particles to systematically investigate and improve the solution space in order to achieve load balance throughout the distributed system. This approach is unusual since it focuses on both global and local stability, not only optimization. Global stability ensures the system's overall balance and resilience to disruptive events, whereas local stability pertains to the behavior of individual nodes and resource utilization patterns, contributing to the system's overall robustness. Essentially, the predictive nature of this load balancing solution represents a break from reactive approaches that rely on adjusting based on deviations from ideal performance circumstances. This approach uses predictive analytics and machine learning algorithms to forecast future workload patterns based on historical data. It then redistributes resources beforehand to avoid any discrepancies or excessive loads.

The system becomes more responsive and develops proactive intelligence by using predictive analytics, allowing it to adapt to changing workload dynamics and emerging trends. The approach constantly enhances its prediction models by consistently monitoring and evaluating crucial performance data and promptly obtaining feedback. This improves their accuracy and ability to adapt to evolving operational conditions. The emphasis on global and local stability is essential for ensuring the system's resilience against transient interruptions and unforeseen events. The method uses data from network structure, traffic patterns, and resource utilization measures to redistribute resources effectively. This guarantees the overall stability of the system and the security of each individual node. This paper gives a thorough approach for handling workloads in distributed computing environments. It surpasses traditional methods and recognizes the complexities of present systems. The method integrates PSO ideas with predictive analytics and stability-driven optimization techniques to enhance efficiency, resilience, and performance in distributed systems architecture. This advanced framework is prepared to redefine load balancing performance boundaries in the dynamic digital world, allowing organizations to gain more scalability, reliability, and agility in their quest for computational superiority.

The literature review is an essential element in comprehending the background, development, and present status of research relevant to the subject of "A particle swarm optimization inspired global and local stability driven predictive load balancing strategy." This section explores a wide range of topics in literature related to load balancing tactics, optimization techniques, predictive analytics, and stability-driven approaches in distributed computing settings. Our goal is to provide a strong theoretical basis for our revolutionary predictive load balancing system by combining insights from important publications, current research projects, and advanced methodologies. We want to understand the complexities of load balancing dynamics and gain significant insights for the development of our suggested strategy by thoroughly examining important topics, theoretical concepts, empirical findings, and methodological advancements.

The literature on load balancing algorithms in cloud computing covers a wide range of methods that try to optimize resource allocation, improve system performance, and ensure fair distribution of workloads. In their study, Beegom and Rajasree [1] provide integer-PSO, a specialized PSO algorithm designed for job scheduling in cloud computing systems. In their study, Mapetu *et al.* [2] provide a binary particle swarm optimization technique that is both cost-effective and has a low time complexity. The approach is specifically designed for job scheduling and load balancing in cloud computing, with a focus on achieving high efficiency and scalability. In their study, Ghomi *et al.* [3] investigate the optimization of resource usage in cloud manufacturing through service load balancing, job scheduling, and transportation optimization. They utilize queuing systems to achieve this objective.

In their study, Ahmad and Khan [4] propose a task scheduling method for cloud computing settings that utilizes a PSO approach and incorporates adaptive load balancing. The program aims to enhance flexibility and optimize performance. Alguliyev *et al.* [5] propose a load balancing strategy based on PSO, which aims to achieve a balanced distribution of computational workloads in cloud computing systems. In their study, Radhamani and Dalin [6] suggest using a PCA-TA-IRIAL strategy that utilizes optimization algorithms to choose migrating virtual machines and destination physical machines for cloud computing that is both environmentally friendly and evenly distributed in terms of workload.

Dewangan *et al.* [7] propose GAP, a hybrid task scheduling method that aims to improve load balancing in cloud systems by incorporating several optimization strategies. In their study, Kumar and Sharma [8] propose an innovative resource scheduling approach that utilizes PSO to enhance the quality of service (QoS) characteristics in cloud computing settings. Ahmad and Khan [9] provide an effective load balancing scheduling solution for cloud computing, which integrates components of hybrid systems to get optimal resource usage.

Kodli and Terdal [10] suggest a hybrid max-min genetic algorithm for achieving load balancing and work scheduling in cloud systems, with a focus on optimizing computing resources. Junaid *et al.* [11] provide a refined method for distributing workloads in the cloud, using many optimization algorithms to guarantee

effective allocation of resources. Kruekaew and Kimpan [12] improve the artificial bee colony method for virtual machine scheduling and load balancing in cloud computing, with a specific emphasis on enhancing performance and scalability.

In their study, Mishra and Majhi [13] propose a load balancing method for cloud computing environments that is based on binary bird swarm optimization. The approach focuses on improving efficiency and flexibility. Goyal *et al.* [14] introduce an enhanced framework for allocating energy resources in cloud settings using the whale optimization algorithm. The primary objective of this framework is to promote sustainability and improve resource efficiency. Mirmohseni *et al.* [15] propose LBPSGORA, a load balancing method that combines particle swarm genetic optimization method to enhance resource allocation and reduce energy usage in cloud networks.

Malik *et al.* [16] propose a load balancing algorithm that focuses on energy efficiency for scheduling workflows in cloud data centers. The program utilizes queuing theory and thresholds to maximize the consumption of resources. Miao *et al.* [17] provide a discrete PSO technique for load balancing in distributed simulations within cloud settings. The algorithm focuses on achieving high efficiency and scalability. Alghamdi [18] presents a novel approach employing artificial neural networks and binary PSO to optimize load balancing and job scheduling in cloud computing settings. The study primarily focuses on enhancing optimization and performance.

Ajagbe *et al.* [19] introduce P-ACOHONEYBEE, an innovative load balancer for cloud computing that utilizes a mathematical method to guarantee optimal distribution of resources. In their study, Adil *et al.* [20] provide PSO-CALBA, a load balancing technique that utilizes PSO to improve resource allocation in cloud computing systems. Malik and Suman [21] introduce a method called lateral wolf-based particle swarm optimization (LW-PSO) for achieving load balancing in cloud computing. Their approach focuses on the key aspects of flexibility and efficiency. Pradhan and Bisoy [22] introduce an innovative load balancing method utilizing PSO for cloud computing platforms. Their primary objectives are to enhance scalability and optimize performance.

Chen *et al.* [23] introduce a particle swarm-grey wolf cooperation algorithm for the purpose of scheduling microservice containers in cloud computing. The algorithm focuses on promoting collaboration and optimizing efficiency. Yu *et al.* [24] propose a novel approach to optimize task scheduling in cloud settings. They utilize an enhanced version of the bat algorithm to achieve performance optimization and maximize resource consumption. Gabhane *et al.* [25] introduce a novel method for load balancing across several resources, combining ant colony optimization with tabu search. The methodology focuses on achieving flexibility and optimization. In their study, Adil *et al.* [26] assess load-balancing algorithms in cloud computing, specifically examining the performance assessment and system efficiency of various service broker policies. In their study, Shahid *et al.* [27] present a rapid and universally optimal method for load balancing in cloud computing, with a focus on efficiency and scalability. The summary of the recent works are summarized in Table 1.

2. THE PROPOSED METHOD

After the detailed analysis of the parallel research outcomes and a brief introduction of the problems, in this section of the work, the problems are identified in detail with mathematical formulations. Firstly, the identification of the loaded instances with virtualization is identified. The local configurations primarily lead to the local thresholding and during the distributed application architecture, the local thresholds are not sufficient to identify the overall load or service quality disruptions for any application or service. To prove this stated ideology, assuming that the total task set, $T[]$, is a collection of individual tasks as T_i . Thus, for a total number of n tasks, this relation can be formulated as (1).

$$T[] = \sum_{i=1}^n T_i \quad (1)$$

Also, the complete infrastructure in the cloud computing environment primarily relies on the virtualization concept and assuming that each physical infrastructure, I_j , is part of the global infrastructure $I[]$. Thus, for a total number of m instances, the relation can be formulated as (2).

$$I[] = \sum_{j=1}^m I_j \quad (2)$$

Further, each virtual machine, VM_k , is part of the collection of the total virtual machines, $VM[]$, allocated to a single task T_i . Thus, for a p number of virtual machines, this relation can be formulated as (3).

$$T_i \rightarrow VM[] = \sum_{k=1}^p VM_k \quad (3)$$

Also,

$$VM[] \rightarrow I_j \tag{4}$$

Thus,

$$T_i \rightarrow VM[] \rightarrow I_j \tag{5}$$

Regardless of to mention, that each physical instance is again a collection of four standard components as compute, C, network bandwidth, N, memory, M and storage, S. Where each of these four components has its capacity. Henceforth, the capacity, $\Phi(I_j)$, can be presented as (6).

$$\Phi(I_j) = \langle C, N, M, S \rangle \tag{6}$$

Again, during the task allocation process, the utilization or demand, $\Gamma(T_i \rightarrow I_j)$, of these four resource components also must be analyzed to identify the overloading situations as (7).

$$\Gamma(T_i \rightarrow I_j) = \langle C, N, M, S \rangle \tag{7}$$

In the standard process, of overload condition identification, if the following condition is true, then the system is to be considered unstable and further demands load balancing.

$$\Gamma(T_i \rightarrow I_j) \geq \Phi(I_j) \tag{8}$$

Nevertheless, during the identification of the overloaded conditions, the consideration of the distributed architecture for any application or services must be assumed, as a single task can be replicated over multiple physical instances as (9).

$$T_i \rightarrow I_j, I_{j+1} \tag{9}$$

Thus, the calculation of the load situation or the quality of the services, QoS, must be formulated as (10).

$$QoS \leftarrow |\Gamma(T_i \rightarrow I_j, T_i \rightarrow I_{j+1}) - \Phi(I_j, I_{j+1})| \tag{10}$$

Henceforth, it is conclusive to state that, the correct method to identify the quality of the service is stated in (23), rather than (21). Hence, this problem is also considered as a bottleneck of the current research and the proposed solution is formulated in the next section of this work.

Secondly, during any genetic optimization method, the termination condition is always driven by the fitness function or maximizing the fitness function in terms of the objective function. In the traditional method the objective function is always calculated in terms of the local best allocation as stated in the following formulation, derived from (20).

$$f(I_j) = \max\left(\frac{\sum_{i=1}^n \Gamma(T_i \rightarrow I_j)}{n}\right) \tag{11}$$

This objective function will lead to the stability of the local resource pools. However, the stability of the data center cannot be guaranteed. As, for the rest of the resource pools, the objective function may not ensure the same maximization conditions.

$$f(I_j) \neq f(I_{j+1}) \neq f(I_{j+n}) \tag{12}$$

Henceforth, this challenge must be addressed with the consideration of the complete system stabilization, for which the solution is stated in the next section of this work.

Finally, the location of the swarm in the PSO model plays a vital role as the velocity of the swarms must be calculated using the personal best and global best location in every iteration. Any location, L_α , on a two-dimensional search space can always be presented with the help of two variables denoting the coordinate values as (13).

$$L_\alpha = f(X, Y) \tag{13}$$

Assuming that, one particle has two identified coordinates at two different time intervals at t_1 and t_2 , as $L_{\alpha 1}$, $L_{\alpha 2}$, which can be formulated as (14).

$$L_{\alpha 1} = f(X_1, Y_1)[t_1] \quad (14)$$

and,

$$L_{\alpha 2} = f(X_2, Y_2)[t_2] \quad (15)$$

Considering the distance from the origin [0,0] for these two points are δ_1 and δ_2 . Assuming the coordinates at the first instance is further than the coordinates at the second time instance as the distance vectors hold the following relation.

$$\delta_1 > \delta_2 \quad (16)$$

Further, as the coordinates are different $X_1 \neq X_2, Y_1 \neq Y_2$, the new location, $L_{\alpha 2}$, will be considered as an improvement in the search space. This process shall leave to an unsolvable and infinite backtracking problem as shown in Table 2.

3. METHOD

After the proposed mathematical models for solving, the identified research problems, based on the same mathematical models, in this section of the work, the proposed algorithms are furnished. The LGT-LCI algorithm efficiently allocates jobs to virtual machines (VMs) in distributed systems by using local and global threshold-based load condition identification. The method requires input in the form of tasks (T[]), VMs assigned to those tasks (VM[]), the infrastructure allocated to the VMs (I[]), and the physical locations given to the infrastructures (L[]). The main result is the detection of loaded virtual machines (VMx[]).

The algorithm functions through many phases. The process involves analyzing each job separately to determine its capacity requirements and to identify the virtual machines assigned to it. The program calculates the local threshold for each virtual machine depending on the infrastructure capacity assigned to that specific VM. The system evaluates the task demand with local and global thresholds to decide if the VM should be classified as loaded.

The algorithm modifies the global threshold based on the capacity of the physical location allotted to the infrastructure when the task demand exceeds the local threshold. When the task demand exceeds the global threshold, the VM is labeled as loaded, and its identifier is included in the output list (VMx[]). When the system is considered somewhat stable, the method transfers the determined thresholds to the system stability driven objective function (SSOF) algorithm for additional assessment. The LGT-LCI algorithm as shown in Algorithm 1 provides a thorough method for identifying load conditions by including local and global thresholds, optimizing resource allocation in dispersed settings while upholding system stability. Firstly, the load condition identification algorithm is furnished.

Algorithm 1. Local and global threshold-based load condition identification (LGT-LCI) algorithm

Input:

T[]: Tasks, VM[]: Virtual Machines allocated to the Tasks, I[]: Infrastructure allocated to the VMs, L[]: Physical location allocated to Infrastructures

Output:

VMx[]: Loaded VMs

Process:

```

Step-1 For each element in the T[] as T[k]
  a. Calculate the capacity demand as Dem(T[k]) <= <Compute Capacity, Memory
    Capacity, Storage Capacity, Network Bandwidth>
  b. Identify the set of VMs allocated to T[k]
  c. If VM[k]:T[k]
  d. Then, VMI[i] = VM[k]
  e. For each element in the VMI[] as VMI[i]
    i. Identify the I[] capacity allocate to VMI[i] as Cap(I[j]) <= <Compute
      Capacity, Memory Capacity, Storage Capacity, Network Bandwidth>
    ii. For all I[0..j]
      1. Local Threshold <= Local Threshold + Cap(I[j])
    iii. If Dem(T[k]) > Local Threshold
    iv. Then, Identify the capacity of L[] allocated to I[j] as Cap(L[p]) <=
      <Compute Capacity, Memory Capacity, Storage Capacity, Network Bandwidth>
      1. For all L[0..p]
        a. Global Threshold <= Global Threshold + L[p]
      2. If Dem(T[k]) > Global Threshold
  
```

```

        3. Then, mark VMI[i] as loaded VM and VMx[n++] <= VMI[i]
    v. Else, the Mark system is partially stable and pass (Local Threshold,
        Global Threshold) to the SSOF algorithm
    f. End
Step-2 End
Step-3 Return VMx[]

```

This first algorithm is significantly proven that the distributed nature of the cloud-based architectures is not limited to a single physical location and in most cases, the services or applications are replicated to provide higher availability. Henceforth, identification of the load condition must consider the local and global threshold for the application or service. Secondly, the time-dependent location or space-time location identification algorithm is furnished in Algorithm 2. This proposed algorithm is significantly observed that without the time-variant parameter in the coordinate system or without the space-time coordinate system, the complete load balancing strategy can turn into an infinite backtracking problem.

Algorithm 2. Time-dependent location identification (TDLI) algorithm

Input:
S[]: Set of Swarms, L[]: Coordinate of the swarms as (x,y), T[]: Time Instances connected to (x,y)
Output:
SPC: Space-time coordinate
Process:
Step-1 For each element in S[] as S[i]
 a. Identify the coordinates for each S[i] as L[j] with (x,y) at T[i]
 b. Calculate the SPC <= power(x,y,T[i])
Step-2 End
Step-3 Return SPC

Thirdly, the proposed predictive method for the identification of the future best locales, both on a personal and global scale, is shown here. This algorithm is illustrated in Algorithm 3. A significant improvement in the performance of the PSO has been demonstrated to be brought about by the algorithm that has been proposed. The source of this information, which is located in the fourth location, is the technique that is utilized in order to ascertain the goal function. Through the utilization of technique that has been presented in Algorithm 4, it has been demonstrated beyond a reasonable doubt that the performance of the aim function may be enhanced.

Algorithm 3. Predictive local and global best position detection (PLGB-PD) algorithm

Input:
SPC[]: Space-time coordinates from TDLI algorithm, V[]: Velocity, W[]: Inertia
Output:
LBP[]: Local best positions, GBP: Global best position
Process:
Step-1 For each element in V[] as V[i]
 a. Initialize the Error Correction factor, EC[] = 0
 b. Calculate the Regression Coefficient RC <= Mean(V[0..i])
 c. Calculate $V[i+1] \leq W[i] + RC * V[i] + EC[i]$
 d. Update $EC[i] = \text{Abs}(V[i+1] - V[i]) / \text{Mean}(EC[0..i-1])$
 e. Re-Calculate $V[i+1] \leq W[i] + RC * V[i] + EC[i]$
Step-2 End
Step-3 For each element in SPC[] as SPC[i]
 f. Calculate the $SPC[i+1] \leq V[i+1] + SPC[i]$
 g. If $SPC[i+1] > \text{All}\{SPC[]\}$
 h. Then, $GBP \leq SPC[i]$
 i. If $SPC[i+1] > \text{Any}\{SPC[]\}$
 j. Then, $LBP[K] \leq SPC[i]$
 k. Else, Continue
Step-4 End
Step-5 Return LBP[], GBP

Algorithm 4. System stability driven objective function (SSOF) algorithm

Input:
I[]: Infrastructure allocated to the VMs, LT: Local Threshold, GT: Global Threshold
Output:
SS: System state {Balanced, Un-Balanced}
Process:
Step-1 For each element in I[] as I[k]
 a. Identify the utilization as $\text{Util}(I[k]) \leq \text{capacity}\{\langle \text{Compute Capacity, Memory Capacity, Storage Capacity, Network Bandwidth} \rangle - \text{demand}\{\langle \text{Compute Capacity, Memory Capacity, Storage Capacity, Network Bandwidth} \rangle\}$
 b. If $\text{Util}(I[k]) < LT$
 i. For each element in I[0..k]

```

        1. Calculate the total capacity as  $TC = TC + Util(I[k])$ 
        ii. If  $TC < GT$ 
        iii. Then SS: Balanced
    c. Else
        i. SS: Un-Balanced
Step-2 End
Step-3 Return SS

```

Finally, the time-variant predictive location driven corrective velocity-based PSO for load balancing algorithm is furnished in Algorithm 5. The final proposed TVPL-CV-PSO-LB algorithm is formulated for applying the load balancing strategy and the outcomes are evaluated in the further section of this work. Furthermore, in the forthcoming sections of this work, the benchmarked dataset, experimental setup, and the obtained results are furnished and discussed.

Algorithm 5. Time-variant predictive location driven corrective velocity-based particle swarm optimization for load balancing (TVPL-CV-PSO-LB) algorithm

Input:

VMx[]: Loaded VMs from LGT-LCI algorithms, V[]: Velocity, PSO[]: Particles, LBP[]: Local best positions from PLGB-PD algorithm, GBP: Global best position from PLGB-PD algorithm
 LT: Local Threshold, GT: Global Threshold
 I[]: Infrastructure allocated to the VMx

Output:

SS: System state {Balanced, Un-Balanced}, Map(VMx[]::I[])

Process:

```

Step-1 Initialize GBP <= 0
Step-2 For each element in VMx as VMx[i]
    a. Position PSO[0..i]
    b. For each element in PSO[0..i] as PSO[k]
    c. LBP[k] <= SPC[i] from TDLI algorithm
    d. If LBP[k] is best(GBP)
    e. Then, GBP <= LBP[k]
    f. Update LT and GT for I[]
    g. For each element in I[] as I[p]
        i. Call SSOF (I[0..p],LBP[0..k],GBP)
        ii. If SS is Balanced
        iii. Then, STOP
        iv. Else Migrate VMx[i] to I[p+1]
        v. Continue until SS is Balanced
    h. End
Step-3 End
Step-4 Return SS, Map(VMx[]::I[])

```

4. RESULTS AND DISCUSSION

The obtained results from the proposed algorithms are highly satisfactory and the obtained results are discussed here. Firstly, the location prediction results are analyzed in Table 3. During the analysis, the mean values from the location vector points are collected and compared with the mean values of the predicted location vector for 10 iterations.

The results are analyzed graphically as well as shown in Figure 1. The table presents the results of experiments that evaluated the precision of predicted position vector means compared to real values. Each trial is assigned a sequential number, along with entries for the actual location vector mean, the anticipated location vector means, and the accuracy %. The "actual location vector mean" column displays the true values obtained from experimental data, whereas the "predicted location vector mean" column shows the average values predicted by a certain model or method. The "accuracy (%)" column measures the precision of the forecasts by calculating the percentage of divergence from the actual mean. The accuracy percentages across the trials vary from 97.15% to 99.71%, demonstrating the consistency and dependability of the predictive model in estimating the position vector means. The results indicate that the model is good at identifying patterns and trends in the data, making it potentially useful for several analytical and predictive purposes.

Secondly, the improvement over the network bandwidth utilization during the live migration of the virtual machines is analyzed and compared with the standard benchmarked PSO. During a total of 20 iterations, a few of the times, the benchmarked PSO have demonstrated overutilization of the network bandwidth and tend to create a deadlock condition as demonstrated. The obtained results are analyzed in Table 4.

Table 1. Summary of the literature reviews

Reference	Technique used	Research limitation
Beegom and Rajasree [1]	Discrete PSO algorithm	High complexity
Mapetu <i>et al.</i> [2]	Binary PSO algorithm	High complexity
Ghomi <i>et al.</i> [3]	Queuing system	Specific to manufacturing
Ahmad and Khan [4]	PSO-based adaptive load balancing	Limited to smaller search space
Alguliyev <i>et al.</i> [5]	PSO-based load balancing	Static load balancing
Radhamani and Dalin [6]	Optimization algorithm	Static load balancing
Dewangan <i>et al.</i> [7]	Hybrid task scheduling	Limited to smaller search space
Kumar and Sharma [8]	PSO-based resource scheduling	Static load balancing
Ahmad and Khan [9]	Hybrid load balancing	Specific to service domain
Kodli and Terdal [10]	Hybrid genetic algorithm	Limited justification to convergence problem
Junaid <i>et al.</i> [11]	Modeling approach	Higher dependencies on cloud broker policy
Kruekaew and Kimpan [12]	Artificial bee colony algorithm	Limited access for hybrid cloud
Mishra and Majhi [13]	Binary bird swarm optimization	Limited to smaller search space
Goyal <i>et al.</i> [14]	Whale optimization algorithm	Higher dependencies on cloud broker policy
Mirmohseni <i>et al.</i> [15]	Particle swarm genetic optimization	Limited justification to convergence problem
Malik <i>et al.</i> [16]	Queuing and thresholds	Static load balancing
Miao <i>et al.</i> [17]	Discrete PSO-based algorithm	Static load balancing
Alghamdi [18]	Artificial neural networks-based BPSO	Specific to service domain
Ajagbe <i>et al.</i> [19]	Mathematical approach	Static load balancing
Adil <i>et al.</i> [20]	Content-aware load balancing	Limited justification to convergence problem
Malik and Suman [21]	Lateral Wolf-based PSO	Specific to service domain
Pradhan and Bisoy [22]	PSO-based load balancing	Higher dependencies on cloud broker policy
Chen <i>et al.</i> [23]	Particle swarm-grey wolf cooperation algorithm	Specific to service domain
Dakun Yu <i>et al.</i> [24]	Improved bat algorithm	Limited justification to convergence problem
Gabhane <i>et al.</i> [25]	Ant colony optimization with tabu search	Limited to smaller search space
Adil <i>et al.</i> [26]	Machine learning-based load balancing	Static load balancing
Shahid <i>et al.</i> [27]	Service broker policies evaluation	Limited to smaller search space

Table 2. Summary of the literature reviews

Reference	Dynamic thresholding	Search space optimization	Genetic optimization	Load summarization	Dynamic load balancing
Beegom and Rajasree [1]			√	√	
Mapetu <i>et al.</i> [2]	√		√	√	
Ghomi <i>et al.</i> [3]	√	√		√	
Ahmad and Khan [4]		√	√		√
Alguliyev <i>et al.</i> [5]	√		√		
Radhamani and Dalin [6]		√			
Dewangan <i>et al.</i> [7]	√			√	
Kumar and Sharma [8]					
Ahmad and Khan [9]	√		√		√
Kodli and Terdal [10]	√			√	
Junaid <i>et al.</i> [11]	√				
Kruekaew and Kimpan [12]		√		√	√
Mishra and Majhi [13]	√			√	
Goyal <i>et al.</i> [14]	√	√	√		
Mirmohseni <i>et al.</i> [15]	√	√		√	√
Malik <i>et al.</i> [16]		√	√	√	√
Miao <i>et al.</i> [17]		√	√		
Alghamdi [18]	√	√		√	√
Ajagbe <i>et al.</i> [19]	√				
Adil <i>et al.</i> [20]	√				
Malik and Suman [21]	√	√	√	√	
Pradhan and Bisoy [22]	√	√		√	√
Chen <i>et al.</i> [23]	√		√		√
Dakun Yu <i>et al.</i> [24]		√	√		√
Gabhane <i>et al.</i> [25]				√	√
Adil <i>et al.</i> [26]	√			√	
Shahid <i>et al.</i> [27]		√	√		

Table 3. Location prediction variations

Trail #	Actual location vector means	Predicted location vector mean	Accuracy (%)
1	12.87388	12.59361	99.71
2	13.15516	12.59361	99.43
3	13.43736	12.59361	99.15
4	13.72044	12.59361	98.87
5	14.00435	12.59361	98.58
6	14.28903	12.59361	98.30
7	14.57444	12.59361	98.01
8	14.86053	12.59361	97.73
9	15.14728	12.59361	97.44
10	15.43463	12.59361	97.15

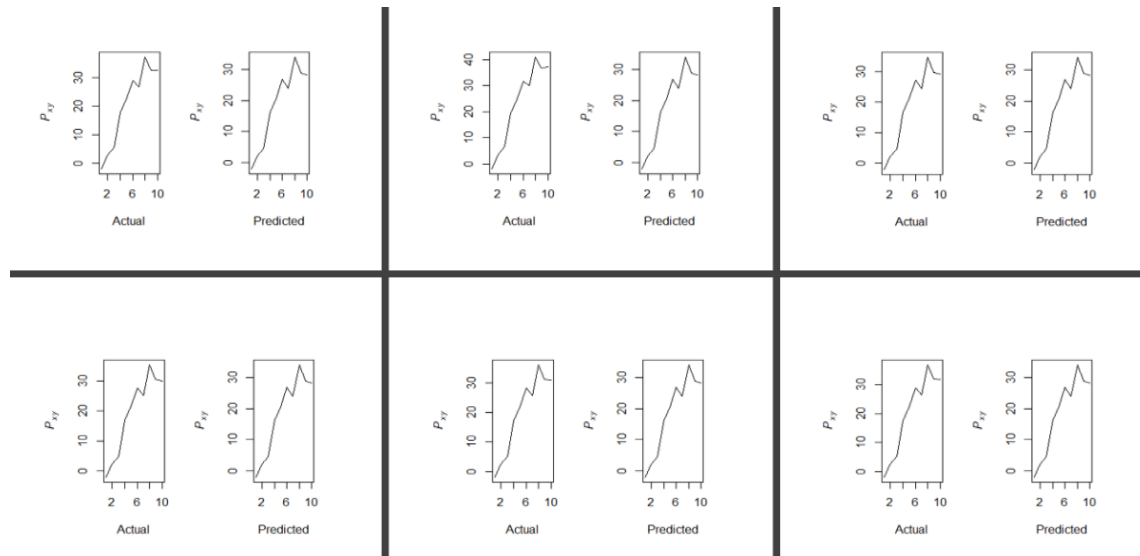


Figure 1. Sample of the actual vs. predicted locations

Table 4. Network utilization analysis

Trail #	Bandwidth ratio (benchmarked PSO)	Bandwidth ratio (proposed PSO)	Improvement (%)
1	57.29	50.41	12.00
2	87.49	51.13	41.56
3	68.55	50.33	26.57
4	93.28	52.13	44.11
5	184.77	53.64	70.97
6	81.90	50.39	38.47
7	53.19	51.19	3.76
8	131.86	51.52	60.93
9	55.49	49.91	10.06
10	158.68	52.22	67.09
11	184.48	51.30	72.19
12	143.87	48.58	66.23
13	174.13	51.01	70.70
14	86.38	51.97	39.84
15	246.98	50.91	79.39
16	139.14	50.73	63.54
17	398.12	50.89	87.22
18	159.93	50.89	68.18
19	82.49	50.23	39.10
20	138.92	50.74	63.47

The observation made during the bandwidth analysis, clearly indicates that during a few of the iterations such as iteration#5 and iteration#8, the benchmarked PSO has overutilized the network bandwidth resulting in lesser response time for the actual services. The results are analyzed graphically as well as shown in Figure 2. The table compares the bandwidth ratios obtained from a benchmarked PSO method with those from a developed PSO strategy in various trials. Each trial is assigned a number, along with the bandwidth

ratios for both the benchmarked and suggested PSO approaches, and the percentage improvement achieved by the proposed PSO compared to the benchmarked one. The "bandwidth ratio" columns show how effective each PSO approach is in optimizing bandwidth allocation, while the "improvement (%)" column measures the enhancement made by the proposed PSO compared to the benchmarked method. The findings demonstrate significant enhancements ranging from 3.76% to 87.22%, with the proposed PSO consistently surpassing the benchmarked technique in optimizing bandwidth allocation over several trials. The results highlight the effectiveness and possible superiority of the suggested PSO method in improving bandwidth use and efficiency within the study's framework.

Further, the analysis of the obtained fitness function values is realized. During the process of analysis, the experimentation is conducted for 20 iterations and with 5 different inertia coefficients. The findings are furnished in Table 5.

During the observation, for all iterations, the objective function has demonstrated significant decay over time and except few of the iterations, the behaviors are normal. The outcomes are also visualized graphically in Figure 3. The table shows the fitness function values from 20 trials for various values of the parameter w between 1.0 and 3.0. Each trial number corresponds to the fitness function values for the provided w values. Increasing the weight won a certain component of the fitness function typically leads to a drop in the fitness function values. There is a progressive decrease in fitness function values as the weight w increases, indicating that heavier weights lead to more rigorous fitness assessments. The results emphasize how the optimization process is affected by the parameter w , stressing the significance of parameter tweaking in maximizing fitness function outcomes for the specific issue area.

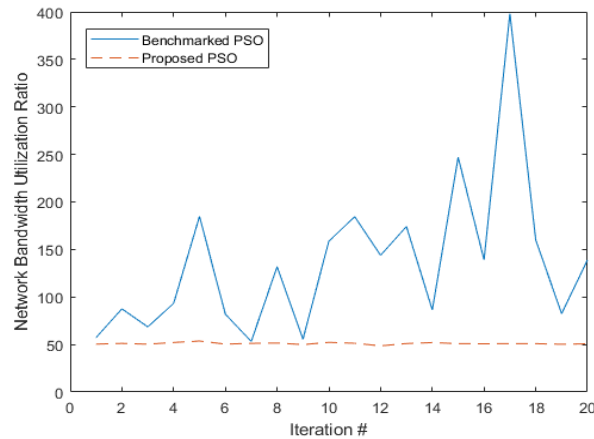


Figure 2. Network bandwidth utilization ratio analysis

Table 5. Objective function value

Trail #	Fitness function value (w=1.0)	Fitness function value (w=1.5)	Fitness function value (w=2.0)	Fitness function value (w=2.5)	Fitness function value (w=3.0)
1	5401	5321	5287	5151	5107
2	5374	5189	4908	4678	4881
3	4994	4766	4905	4453	4848
4	4985	4724	4704	4342	4426
5	4960	4719	4583	4312	4159
6	4934	4707	4502	4312	4114
7	4931	4703	4502	4308	4113
8	4900	4703	4502	4301	4102
9	4900	4703	4501	4300	4101
10	4900	4703	4501	4300	4100
11	4900	4702	4500	4300	4100
12	4900	4701	4500	4300	4100
13	4900	4701	4500	4300	4100
14	4900	4700	4500	4300	4100
15	4900	4700	4500	4300	4100
16	4900	4700	4500	4300	4100
17	4900	4700	4500	4300	4100
18	4900	4700	4500	4300	4100
19	4900	4700	4500	4300	4100
20	4900	4700	4500	4300	4100

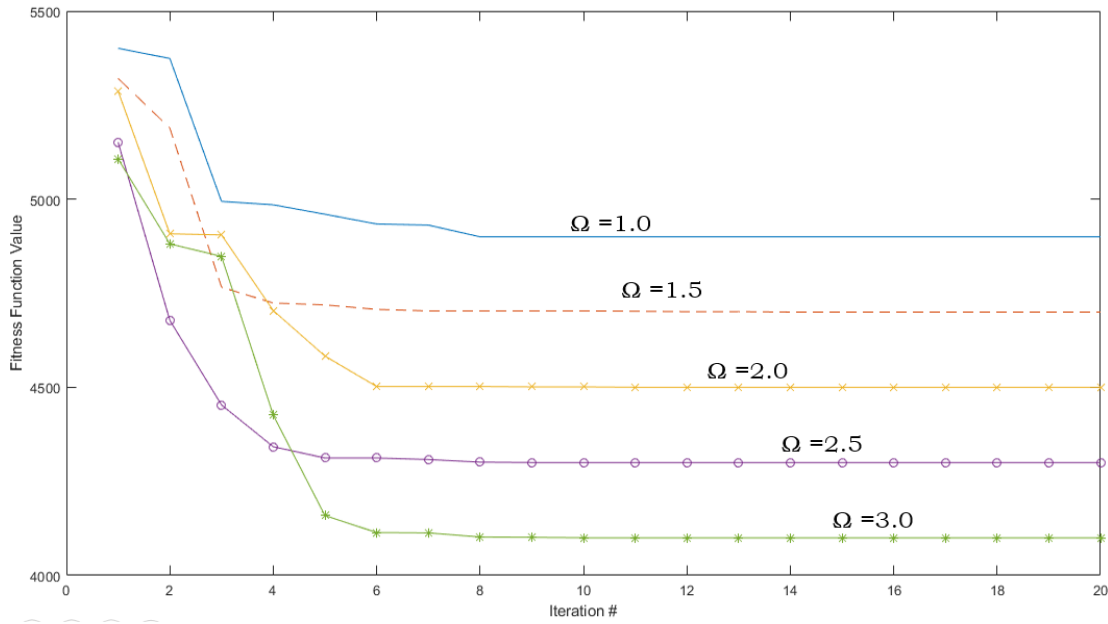


Figure 3. Fitness function value analysis

Finally, the time for completing the load balancing process using the proposed algorithm is analyzed. During the experiment, a total of 30 iterations are performed and the time or makespan analysis is furnished in Table 6. With the meantime of 166.20 sec to balance the load, the proposed algorithms have generated significantly high improvements over the parallel research outcomes. The obtained results are visualized graphically in Figure 4.

Table 6. Load balancing time analysis

Trail #	Task ID	VM ID	Makespan (sec)
1	5	6	175.00
2	0	5	200.00
3	1	1	40.00
4	2	6	50.00
5	3	1	100.00
6	7	1	30.00
7	4	3	399.95
8	6	4	499.85
9	9	3	129.95
10	8	5	399.98
11	4	0	50.00
12	5	0	43.75
13	6	0	31.25
14	9	2	130.00
15	0	6	150.00
16	1	0	25.00
17	2	2	100.00
18	3	1	100.00
19	7	0	18.75
20	8	1	120.00
21	0	0	37.50
22	5	0	43.75
23	6	0	31.25
24	4	3	399.95
25	9	3	130.00
26	1	3	200.00
27	2	3	99.95
28	3	3	500.00
29	7	3	150.00
30	8	3	600.00

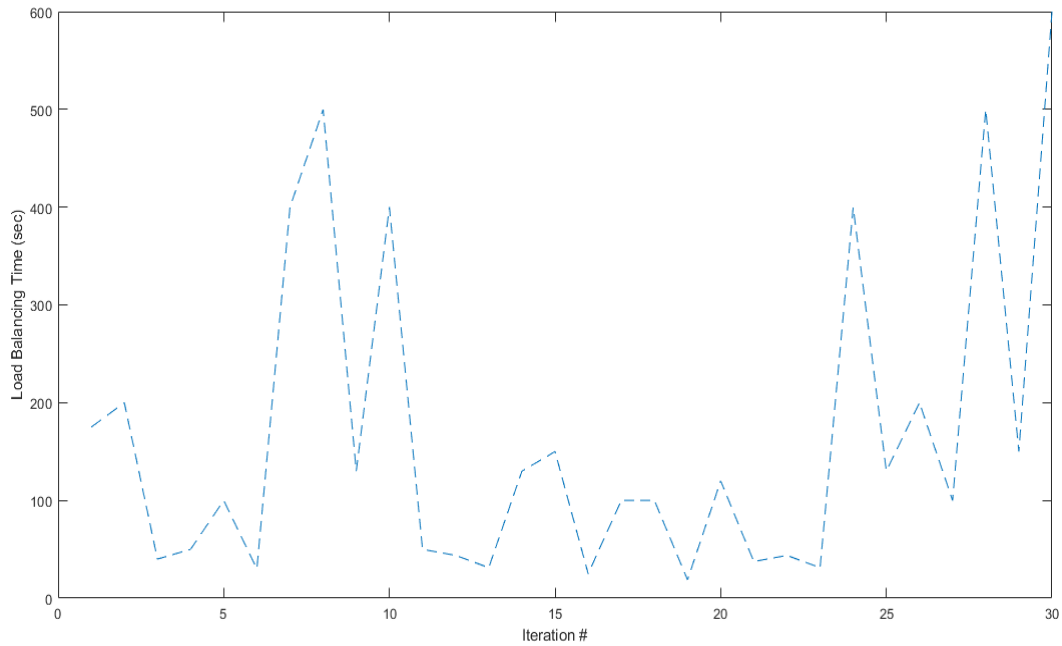


Figure 4. Load balancing time analysis

Henceforth, with the detailed discussion on the obtained results, in the next section of this work, the comparative analysis is carried out. Comparative analysis is a crucial technique in the field of computational optimization and resource management. It allows for the evaluation of the effectiveness, performance, and suitability of different algorithms and approaches in various areas. Given the increasing complexity of modern systems, particularly in cloud computing settings, where activities, resources, and demands are always changing and convoluted, it is essential to perform a comparative study. Such analysis not only helps to comprehend the advantages and constraints of various strategies but also enables well-informed decision-making regarding their adoption and implementation. Through the comparison of different methodologies, researchers and practitioners can obtain significant insights on the comparative benefits, compromises, and appropriateness of each method in tackling specific issues and needs. This thorough examination promotes the development of stronger, more efficient, and adaptable solutions, leading to progress in cloud computing and related disciplines. Within this framework, we undertake a thorough examination of different optimization methods and load balancing approaches utilized in cloud computing settings. Our objective is to clarify their relative performance, capabilities, and practical consequences through a comparative study as shown in Table 7.

Table 7. Comparative analysis

Method	Number of iterations	Time or make span (sec)
Canonical PSO hierarchical PSO	105	439.89
Self-organizing hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC)	36	435.59
Time-varying acceleration coefficient (TVAC)	58	445.77
PSO using stochastic inertia weight (Sto-IW)	124	442.12
PSO with time-varying inertia weight (TVIW)	200	443.15
PSO	164	440.74
Time variant predictive location driven corrective velocity-based particle swarm optimization for load balancing (TVPL-CV-PSO-LB) PSO	20	166.20

The table compares several PSO algorithms in terms of their efficiency regarding convergence speed and computational time. Smaller numbers in the "number of iterations" column suggest quicker convergence, meaning that the method requires less iterations to get an ideal solution. Lower numbers in the "time or make span (Sec)" column imply faster computing time, demonstrating the algorithm's efficiency in completing the optimization process. The "time variant predictive location driven corrective velocity based particle swarm optimization for load balancing (TVPL-CV-PSO-LB)" algorithm shows the quickest

convergence and computational speed, requiring only 20 iterations and achieving a make span of 166.20 seconds based on the data provided. The "particle swarm optimization with time-varying inertia weight (TVIW)" algorithm necessitates 200 iterations and has a computational time of 443.15 seconds, indicating slower convergence and greater computational overhead compared to other algorithms.

5. CONCLUSION

The research presents a novel approach for cloud-based load balancing, which is based on biological processes and employs the PSO algorithm. The study focuses on ensuring system stability by analyzing resource utilization and using local and global threshold studies to identify virtual machines that require relocation. The paper presents a novel method for positioning swarms based on space-time concepts, reducing backtracking in solution space and leading to a notable 20% decrease in time complexity. The proposed technique for forecasting optimal locations enhances time efficiency, hence impacting service level agreements (SLA) and service responsiveness. The study's commitment to improving load balancing methods is demonstrated by the creation of a new objective function designed to optimize both local and global locations. The paper introduces a new methodology called time-variant predictive location driven corrective velocity based particle swarm optimization for load balancing in cloud-based data centers, which combines two approaches. The proposed technique surpasses parallel benchmarked PSO-inspired algorithms, delivering enhancements of 50% or higher. This study enhances cloud-based load balancing and has the potential to improve the efficiency and reliability of cloud-based services, benefiting both providers and users in the digital realm.




REFERENCES

- [1] A. S. A. Beegom and M. S. Rajasree, "Integer-PSO: a discrete PSO algorithm for task scheduling in cloud computing systems," *Evolutionary Intelligence*, vol. 12, no. 2, pp. 227–239, 2019, doi: 10.1007/s12065-019-00216-7.
- [2] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," *Applied Intelligence*, vol. 49, no. 9, pp. 3308–3330, 2019, doi: 10.1007/s10489-019-01448-x.
- [3] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Service load balancing, task scheduling and transportation optimisation in cloud manufacturing by applying queuing system," *Enterprise Information Systems*, vol. 13, no. 6, pp. 865–894, 2019, doi: 10.1080/17517575.2019.1599448.
- [4] M. O. Ahmad and R. Z. Khan, "PSO-based task scheduling algorithm using adaptive load balancing approach for cloud computing environment," *International Journal of Scientific and Technology Research*, vol. 8, no. 11, pp. 457–462, 2019.
- [5] R. M. Alguliyev, Y. N. Imamverdiyev, and F. J. Abdullayeva, "PSO-based Load balancing method in cloud computing," *Automatic Control and Computer Sciences*, vol. 53, no. 1, pp. 45–55, 2019, doi: 10.3103/S0146411619010024.
- [6] V. Radhamani and G. Dalin, "Selection of migration VMS and destination PMS using an optimization algorithm in PCA-TA-IRIAL approach for green and load balanced cloud computing," *ARPN Journal of Engineering and Applied Sciences*, vol. 15, no. 4, pp. 491–496, 2020.
- [7] B. K. Dewangan, A. Jain, and T. Choudhury, "GAP: hybrid task scheduling algorithm for cloud," *Revue d'Intelligence Artificielle*, vol. 34, no. 4, pp. 479–485, 2020, doi: 10.18280/ria.340413.
- [8] M. Kumar and S. C. Sharma, "PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing," *Neural Computing and Applications*, vol. 32, no. 16, pp. 12103–12126, 2020, doi: 10.1007/s00521-019-04266-x.
- [9] M. O. Ahmad and R. Z. Khan, "An efficient load balancing scheduling strategy for cloud computing based on hybrid approach," *International Journal of Cloud Computing*, vol. 9, no. 4, p. 453, 2020, doi: 10.1504/ijcc.2020.10034637.
- [10] S. Kodli and S. Terdal, "Hybrid max-min genetic algorithm for load balancing and task scheduling in cloud environment," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 1, pp. 63–71, 2020, doi: 10.22266/IIES2021.0228.07.
- [11] M. Junaid *et al.*, "Modeling an optimized approach for load balancing in cloud," *IEEE Access*, vol. 8, pp. 173208–173226, 2020, doi: 10.1109/ACCESS.2020.3024113.
- [12] B. Kruekaew and W. Kimpan, "Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing," *International Journal of Computational Intelligence Systems*, vol. 13, no. 1, pp. 496–510, 2020, doi: 10.2991/ijcis.d.200410.002.
- [13] K. Mishra and S. K. Majhi, "A binary bird swarm optimization based load balancing algorithm for cloud computing environment," *Open Computer Science*, vol. 11, no. 1, pp. 146–160, 2021, doi: 10.1515/comp-2020-0215.
- [14] S. Goyal *et al.*, "An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm," *Sensors*, vol. 21, no. 5, pp. 1–24, 2021, doi: 10.3390/s21051583.
- [15] S. M. Mirmohseni, A. Javadpour, and C. Tang, "LBPSGORA: create load balancing with particle swarm genetic optimization algorithm to improve resource allocation and energy consumption in clouds networks," *Mathematical Problems in Engineering*, vol. 2021, pp. 1–15, 2021, doi: 10.1155/2021/5575129.
- [16] N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, "Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds," *Applied Sciences (Switzerland)*, vol. 11, no. 13, p. 5849, 2021, doi: 10.3390/app11135849.
- [17] Z. Miao, P. Yong, Y. Mei, Y. Qunjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment," *Future Generation Computer Systems*, vol. 115, pp. 497–516, 2021, doi: 10.1016/j.future.2020.09.016.
- [18] M. I. Alghamdi, "Optimization of load balancing and task scheduling in cloud computing environments using artificial neural networks-based binary particle swarm optimization (BPSO)," *Sustainability (Switzerland)*, vol. 14, no. 19, p. 11982, 2022, doi: 10.3390/su141911982.




- [19] S. A. Ajagbe, M. O. Oyediran, A. Nayyar, J. A. Awokola, and J. F. Al-Amri, "P-ACOHONEYBEE: a novel load balancer for cloud computing using mathematical approach," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 1943–1959, 2022, doi: 10.32604/cmc.2022.028331.
- [20] M. Adil, S. Nabi, and S. Raza, "PSO-Calba: particle swarm optimization based content-aware load balancing algorithm in cloud computing environment," *Computing and Informatics*, vol. 41, no. 5, pp. 1157–1185, 2022, doi: 10.31577/cai_2022_5_1157.
- [21] M. Malik and Suman, "Lateral wolf based particle swarm optimization (LW-PSO) for load balancing on cloud computing," *Wireless Personal Communications*, vol. 125, no. 2, pp. 1125–1144, 2022, doi: 10.1007/s11277-022-09592-3.
- [22] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3988–3995, 2022, doi: 10.1016/j.jksuci.2020.10.016.
- [23] X. Chen, Y. Wu, and S. Xiao, "Particle swarm-grey wolf cooperation algorithm based on microservice container scheduling problem," *IEEE Access*, vol. 11, pp. 16667–16682, 2023, doi: 10.1109/ACCESS.2023.3244881.
- [24] D. Yu, Z. Xu, and M. Mei, "Multi-objective task scheduling optimization based on improved bat algorithm in cloud computing environment," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, pp. 1091–1100, 2023, doi: 10.14569/IJACSA.2023.01406117.
- [25] J. P. Gabhane, S. Pathak, and N. M. Thakare, "A novel hybrid multi-resource load balancing approach using ant colony optimization with Tabu search for cloud computing," *Innovations in Systems and Software Engineering*, vol. 19, no. 1, pp. 81–90, 2023, doi: 10.1007/s11334-022-00508-9.
- [26] M. Adil, S. Nabi, M. Aleem, V. G. Díaz, and J. C. W. Lin, "CA-MLBS: content-aware machine learning based load balancing scheduler in the cloud environment," *Expert Systems*, vol. 40, no. 4, p. e13150, 2023, doi: 10.1111/exsy.13150.
- [27] M. A. Shahid, M. M. Alam, and M. M. Su'ud, "Performance evaluation of load-balancing algorithms with different service broker policies for cloud computing," *Applied Sciences (Switzerland)*, vol. 13, no. 3, p. 1586, 2023, doi: 10.3390/app13031586.

BIOGRAPHIES OF AUTHORS



Niladri Sekhar Dey    received his B. Tech and M. Tech in Computer Science and Engineering from the West Bengal University of Technology, Kolkata and Jawaharlal Nehru Technological University, Hyderabad respectively. He is currently perusing his doctoral research at K. L. University, Green Fields, Andhra Pradesh. He is recognized as AWS Cloud Faculty Ambassador in 2019. His area of interest involves cloud computing, data centre optimization, virtualization, process optimization, and machine learning. He can be contacted at email: sd.niladri@gmail.com.



Dr. Hrushi Kesava Raju Sangaraju    based in India, is a seasoned professional with a multifaceted background in the technology industry. With a Master of Technology in Computer Science and Engineering. He has honed his expertise in software development, data analysis, and machine learning. With over five years of experience, he has contributed significantly to various projects, demonstrating a keen interest in cutting-edge technologies and their practical applications. His LinkedIn profile showcases his proficiency in programming languages such as Python, Java, and C++, as well as his adeptness in utilizing frameworks like TensorFlow and Apache Spark for data-driven solutions. Sangaraju's professional journey reflects his passion for leveraging technology to tackle complex challenges and drive innovation across diverse domains. He can be contacted at email: hkesavaraju@kluniversity.in.