International Journal of Basic and Applied Sciences, 14 (7) (2025) 535-547

International Journal of Basic and Applied Sciences

Website: www.sciencepubco.com/index.php/IJBAS
https://doi.org/10.14419/9zdfrx34
Research paper

Multi-Objective Load Balancing in Heterogeneous Distributed Computing Systems Using Particle Swarm Optimization

Vidya S. Handur [1] *, Santosh L. Deshpande [2]

[1] *KLE Technological University, Department of Computer Science and Engineering, Karnataka, India*
[2] *Visvesvaraya Technological University, Department of Computer Science and Engineering, Karnataka, India*
**Corresponding author E-mail: vidya_handur@kletech.ac.in*

Received: October 30, 2025, Accepted: November 14, 2025, Published: November 24, 2025

Abstract

Load balancing distributes incoming traffic across a pool of resources to improve system performance and prevent overutilization or underutilization of resources. In heterogeneous distributed computing systems, dynamic load balancing involves continuous monitoring and redistribution of tasks among available nodes, considering their diverse processing capabilities. As a result, efficient task distribution remains a major concern in these systems. The study proposes an approach that combines probability-based load migration and weighted-based task reallocation using Particle Swarm Optimization (PSO). The primary goal is to minimize task execution time while maximizing overall resource utilization in a heterogeneous environment, with a focus on optimizing memory and CPU usage. The proposed dynamic algorithm effectively adapts to handle numerous tasks and fluctuating nodes. Simulations are conducted using CloudSim. The performance is compared with existing PSO variants: Time-Conscious PSO, PPTS-PSO, EASA-MORU, and HEPGA. Results show that the proposed method significantly outperforms other existing approaches. Specifically, the algorithm achieves an average resource utilization of 98%, reduces execution time, improves throughput, and ensures fair workload distribution across all nodes. The proposed method is highly suitable for distributed computing environments characterized by dynamic task demands and varying computational resource requirements.

Keywords: *Dynamic load balancing; Fairness Index; Makespan; Particle Swarm Optimization; Resource Utilization; Throughput.*

1. Introduction

With the growing demand for network communication and its diverse applications, there is an increasing need for higher data rates and reduced processing and response times within distributed systems. A Distributed Computing System (DCS) comprises autonomous computational nodes that communicate with neighboring nodes to execute tasks effectively [1]. Applications leveraging distributed systems include cloud computing, automated banking systems, and the World Wide Web (WWW), among others. Such applications require optimal utilization of available resources, emphasizing resource sharing, a fundamental characteristic of DCS [2]. Achieving scalability and efficient resource sharing is critical in achieving common objectives, which typically involve maximization of throughput and resource utilization, minimization of response time, and execution time. To meet these goals, workloads must be redistributed among nodes to ensure optimal task allocation and enhanced system efficiency, a process known as load balancing. Load balancing is typically initiated when nodes in a system get either overloaded or underloaded. Various methods have been proposed to address load balancing challenges, with swarm intelligence-based metaheuristics playing a prominent role due to their inherent properties of self-organization, environmental adaptability, and decentralization [3]. Consequently, these techniques are suitable for distributed computing environments. Fig.1 shows some of the applications of Swarm Intelligence.

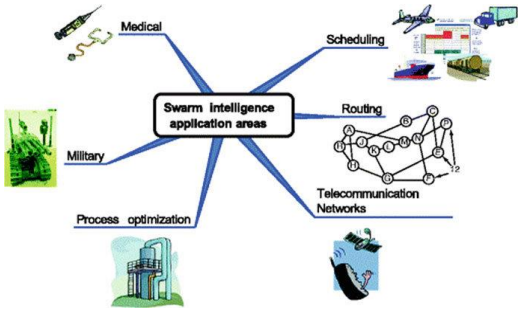


Fig. 1: Swarm Intelligence Applications.

Copyright © Vidya S. Handur, Santosh L. Deshpande. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

There are multiple approaches to load balancing [4]. They are broadly characterized into two: static and dynamic methods [5]. Static load-balancing techniques require prior knowledge of incoming traffic, which is challenging to obtain when the traffic is continuously fluctuating. Handling this type of traffic requires robust load-distribution mechanisms that operate adaptively and without dependency on predefined traffic characteristics. Although traditional methods can provide dynamic solutions, they are expensive and may perform poorly under changing loads [6]. Swarm Intelligence (SI) based solutions can adapt to such changing traffic, leading to optimal results in decentralized systems [7]. Several SI-based algorithms are used for load balancing. Some of these methods include Particle Swarm Optimization [8]-[11], Artificial Bee Colony Optimization [12], Grey Wolf Optimization, Spider Monkey Optimization, Simulated Annealing [13], and Ant Colony Optimization [14]-[16]. The proposed study focuses on how the PSO technique can dynamically balance the load in a heterogeneous distributed computing system.

Load balancing is an NP-hard [17] problem, making exact task-to-resource allocation difficult in dynamic systems. To address this issue, different techniques such as hybrid solutions, heuristics, and meta-heuristics are employed to provide dynamic allocation of tasks to resources. Various studies exist for load balancing. However, with the advancement in technology and the increase in the number of users and their varying demands, load balancing remains a challenging issue. This study emphasizes a novel probability and weighted-based multi-objective load balancing in a heterogeneous distributed computing system using PSO. Compared with other swarm intelligence approaches, PSO uses fewer parameters, is simple to implement, readily parallelizes, and offers effective global exploration. PSO reduces complexity and converges quickly to high-quality solutions. The major contribution of the proposed study lies in the process of load balancing, which involves detecting overload, estimating load, calculating processing times, and migrating tasks to an underutilized node. The process is further optimized to determine the best possible allocation of tasks to nodes using PSO.

The structure of the paper is organized as follows: Section 2 presents previous work related to the proposed study, Section 3outlines the System model, and Section 4 details the proposed methodology. Section 5 describes the optimization model for load balancing, while Section 6 discusses the simulation setup. The results and their discussions are presented in Section 7, and the paper is concluded in Section 8.

2. Related Work

Load balancing is a procedure that refers to the redistribution of load such that nodes in the system are neither overloaded nor underloaded. Numerous studies address load balancing. This section details some research on dynamic load balancing methods.

In [18], a novel load-balancing method using a cuckoo search algorithm was proposed to reduce the response time by redistributing tasks to virtual machines. The algorithm is evaluated against Simulated Annealing, Genetic Algorithm, and Round Robin; results show that Cuckoo Search outperforms all others. In [19], a two-stage cloud-based load-balancing method was proposed. In the initial stage, the basic PSO algorithm is applied, and the results are then used as the initial population in the next stage. The study is deterministic in which tasks are allocated to a Virtual Machine (VM) that is lightly loaded, considering the processing capability of the virtual machine, and with maximum probability. Performance metrics measured through the study are resource utilization and makespan.

In [20], each particle represents a possible solution for assigning tasks to the machine. Each particle vector of size N represents the number of tasks, with the value inside the vector being a random number to signify the resource. To enhance performance and achieve a balance between local and global searches, the inertia weight is adjusted. The performance metrics evaluated include flowtime and makespan. In [21], a hybrid approach combining honey bee behaviour and PSO is proposed. The foraging behaviour of honey bee optimization is utilized, along with resource awareness, to manage resources effectively. Experimental results demonstrate that the proposed technique reduces total processing time, makespan, and degree of imbalance. The hybrid approach surpasses the standard PSO in terms of load balancing performance.

In [22], a PSO-based load balancing was proposed to minimize task transfer time and task execution time. The proposed methodology uses the weight coefficients to indicate the task scheduling significance. By changing these weight coefficients, a better optimality of the target function is achieved. In this methodology, tasks represent the particles, and the virtual machine numbers indicate the particles' position, and continuous values are converted to discrete using the Small Position Value rule.

In [23], a methodology was proposed to increase resource utilization and decrease makespan. The tasks are independent and non-preemptive, indicating the particles' N-dimensional vector. The values of the vector represent the index of the machine on which the task is likely to be executed. The fitness function is calculated using measures of the makespan and resource utilization of each machine. The position vector is updated using a sigmoid function. Balancing is performed until there are no machines that are either underloaded or overloaded, and the standard deviation of the system load is computed. In [24], the proposed task scheduling using swarm intelligence aims to predict system functionality, which is less costly in implementation and requires less time. Therefore, this improves power consumption and helps minimize overloads. The combined method of PSO and Bat optimization algorithms is used for task scheduling.

In [25], the authors proposed adaptive PSO-based task scheduling to reduce task execution time, thereby increasing throughput and resource utilization. The study employed the Adaptive Inertia Weight and Linearly Descending method to achieve an effective balance between local and global searches. Makespan and throughput have been compared with five other PSO-based inertia weight strategies. Experimental results show 60% and 12% gains in average resource utilization and throughput, with a 10% reduction in makespan.

In [26], the authors proposed a novel bio-inspired method called Inventive Particle Swarm Optimization, which efficiently schedules users' tasks to uniformly distribute the load among virtual machines. Additionally, they introduced an algorithm known as Merge Sort with Divide and Conquer. This method allocates resources in the cloud dynamically and more efficiently. The experimentations are carried out in the CloudSim simulator. The results of experiments have proved that the algorithms under study yield good response time, reduced execution time, and improved VM utilization.

In [27], a novel weight assignment method is introduced to balance the load. The proposed algorithm experiments have shown improved performance of three-tier cloud-based web applications, reducing resource failures, single points of failure, slow uncertainty sensing, and unnecessary re-routing. The algorithm measures five server metrics—network buffer occupancy, thread count, CPU utilization, bandwidth utilization, and RAM—to distribute workload and maintain low response time under randomly arriving user requests without degrading performance. In [28], the authors have analyzed various virtual machine load-balancing algorithms for cloud computing. A novel virtual machine load-balancing algorithm was introduced within an Infrastructure as a Service framework and implemented in the Cloud Analyst virtual machine environment. This algorithm demonstrated improved response times and processing times.

In [29], load balancing using modified PSO task scheduling is proposed. In this method, nonpreemptive tasks are scheduled over the available heterogeneous resources. Resource utilization and makespan are measured. In the study, each particle is a task assigned to a virtual machine (VM), represented by position, and the exchanged information is denoted by velocity. Before assigning tasks to VM, the

PSO technique is applied to obtain each particle's best position. To allocate tasks to the least loaded VM, the algorithm continuously checks the existing load status for each VM.

In [30], the authors propose an approach for full utilization of virtual machines and to minimize makespan to accomplish initial load balancing. The SPSO-TCS technique combines Time-Conscious Scheduling with Supportive Particle Swarm Optimization to achieve the goal. The method is divided into two stages. The first stage employs PSO to schedule the tasks on virtual servers. The second stage has a time-aware scheduler to allocate the jobs to virtual servers and minimize makespan. The method computes a balanced schedule initially, but does not continuously re-optimize once the execution begins. In an environment where new tasks arrive dynamically or a few nodes are slowing down, the algorithm would not inherently adjust the schedule.

In [31], the Hybrid HEFT-PSO-GA algorithm (HEPGA) combines Heterogeneous Earliest Finish Time, PSO, and GA, which aims to allocate tasks to resources efficiently. The algorithm integrates PSO and GA to optimize the scheduling of tasks in cloud computing environments. The approach uses the parallel processing capabilities of resources to further enhance resource utilization. The weights of the fitness function are varied, and performance is measured in terms of makespan and Resource Utilization, showing that the approach is adaptable to varying priorities. In this method, some of the fastest nodes are heavily loaded while slower nodes may be underutilized. The method is complex due to the runtime overhead for a larger set of tasks.

In [32], the authors present PPTS-PSO, a hybrid approach that integrates two efficient algorithms to optimize the scheduling of interdependent cloud tasks. This work adopts a hybrid approach, combining an intelligent PSO variant: neighbourhood PSO with the PPTS heuristic. The approach allocates tasks to the most appropriate cloud VM, thereby achieving lower execution times and cost. The method schedules the functions in a batch manner; however, some computational nodes may remain underutilized.

In [33], the authors proposed a method, EASAMORU, to maximize the makespan and use the resources effectively in cloud infrastructure. The technique uses the Dung Beetle Optimization (DBO) to schedule the tasks. The technique effectively balances the load and distributes the resources based on the demands of the cloud infrastructure. In this method, once the schedule is produced by the algorithm, the solution is static. If the node's performance changes or if new tasks arrive, the method would need to re-run the optimization – it doesn't inherently support on-the-fly adjustments.

In [34], the tasks are prioritized based on their features. The virtual machines are given priority levels and categorized based on their importance and availability using the K-means algorithm. The results measured are makespan, energy consumption, load balancing, and resource utilization. In [35], the authors proposed a bandwidth-aware and energy-efficient multi-objective methodology using a genetic algorithm to minimize makespan, network congestion, and energy consumption. The simulations are carried out and tested for data-intensive applications.

3. System Model

In a heterogeneous distributed computing architecture, a server handles many client requests. The clients generate requests that are to be distributed among the servers based on their computational capabilities. Given a system consisting of m computational nodes, denoted as M_1, M_2, ..., M_m, and n' heterogeneous tasks, denoted as T_1, T_2, ..., T_n, with varying lengths l_1, l_2, ..., l_n, where $n >> m$, the tasks must be mapped to computational nodes. Each computational node is uniquely identified by an identity number, a storage capacity 'C' that indicates the maximum queue length, and a service rate measured in Million Instructions Per Second (MIPS). The system's heterogeneity is characterized by differences in its computational capabilities, including server processing speed and queue length. The tasks are characterized by a unique identity number, length in Million Instructions (MI), and preemption type. The study aims to determine an optimal allocation of 'n' tasks to m ' nodes.

4. Proposed Methodology

The proposed methodology is a hybrid approach that combines probability-based and weighted-based load balancing for fair distribution of tasks using Particle Swarm Optimization in a heterogeneous network. The objectives are to maximize resource utilization and minimize the overall execution time of the tasks. It comprises the following steps:

1) Load Estimation
2) Probability-based load balancing
3) Weight-based reallocation of tasks.

Table 1 refers to the key notations used in the study.

Table 1: Key Notations

Symbol	Description
n	Total number of tasks
m	Total number of nodes
T_x	Task number
l_x	Task length
TL_x	Total load on a node X
ECT	Expected Completion Time
TECT	Total Expected Completion Time
μ_x	Processing Speed of a node
TL_{max}	Maximum load in the system
TL_{min}	Minimum load in the system
MS_x	Makespan of node X
t_{min}	Minimum threshold value
t_{max}	Maximum threshold value
RU	Resource Utilization
W_x	Weight of a node X
x	Particle position
v	Particle velocity

4.1. Load estimation

Before load balancing is performed, it is necessary to check for load imbalance in the network. The load is calculated as the length of tasks waiting in the queue for execution. If the length exceeds the preset threshold value, then there is an imbalance, and the balancing of the load is initiated by the node as follows:
If the sum of the completion time of the existing load and the newly assigned task exceeds the expected processing time after migration, the task is transferred to the neighbouring node Mj
Let T_{new} denote the new task with length l_{new} arriving at a node M_i.
The total load on the node M_i after the arrival of the new task is as given in Eq. (1)

$$TL_i = (l_1 + l_2 + \ldots + l_x) + l_{new} \quad (1)$$

Where l_i denotes the length of all the tasks assigned to a node, and TL_i denotes the Total Load on node M_i after adding the length of the newly arriving task.
If TL_i exceeds the threshold, the node has to initiate load balancing.
To calculate the Expected Completion Time, let 'n' be the number of tasks received by the system dynamically and let l_i be the length of the task; then the total load on each node M_j is as given in Eq. (2).

$$TL_j = \sum_{i=1}^{n} x_{ij} l_i \quad \forall j=1,2,\ldots m \quad (2)$$

- l_i is the length of the task,
- x_{ij} is 1 if task 'i is assigned to node 'j', otherwise 0
- and TL_j is the total load on node M_j

The time taken to execute all tasks locally on node M_i is computed as shown in Eq. (3)

$$ECT_i = \frac{TL_i}{\mu_i} \forall i = 1,2,\ldots m \quad (3)$$

Where ECT_i denotes the Expected Completion Time on node M_i, μ_i is the service rate of node M_i.
The total time to execute the tasks on m nodes is as given in Eq. (4)

$$TECT = \sum_{i=1}^{m} ECT_i = \sum_{i=1}^{m} \frac{TL_i}{\mu_i} \quad (4)$$

Where TECT is the Total Expected Completion Time
The Average time to complete the execution of all tasks on m nodes in a fully distributed heterogeneous system is given in Eq. (5)

$$AvgECT = \frac{TCET}{m} \quad (5)$$

The objective of the study is to minimize overall execution time, thereby minimizing the Average Expected Completion Time. The objective function is stated as in Eq. (6)

$$MinimizeAvgECT\ (TL_1, TL_2, \ldots, TL_m) \quad (6)$$

Subject to constraints $t_{min} \leq TL_i \leq t_{max}$, $\forall$ i = 1, …, m, where t_{min} and t_{max} are the minimum and maximum threshold values set on the queue lengths of the nodes.
The threshold value is used as a measure to categorize the nodes as overloaded or underloaded.
Overloaded nodes: A node is considered overloaded if its queue length exceeds the threshold value t_{max}.
Underloaded nodes: A node is considered underloaded if its queue length falls below the minimum threshold value t_{min}.
Fig. 2 illustrates the optimization model proposed for the allocation of tasks to computational nodes.

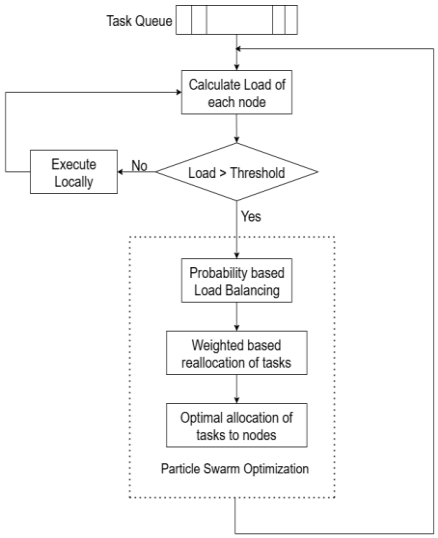


Fig. 2: Optimization Model for Load Balancing Solved by Using PSO.

4.2. Probability-based load balancing

As system load increases, imbalance becomes more likely and must be addressed to maintain the system's performance. An imbalance is a situation in which the queue length either exceeds or falls below some specified threshold value. To balance the load, some of the tasks from the overloaded nodes have to be migrated to suitable nodes.
If a task Ti is to be migrated, it is assigned to a new node Mj with the highest probability. The load is balanced by migrating the tasks probabilistically by considering the previous allocations of tasks to computational nodes, thereby enhancing the resource utilization.
Task migration probability calculation :
Each task T_i is scheduled on a lower-utilized node based on the probability $P_{migrate}$ calculated as the difference ratio, as shown in Eq. (7).

$$P_{migrate} = \frac{TL_{max} - TL_{min}}{TL_{max}} \tag{7}$$

Where TL_{max} and TL_{min} represent the maximum load and the minimum load in the system. The task is migrated from a node M_{max} with maximum load to a node with M_{min} based on the probability $P_{migrate}$ and generating a random number. If the generated random number is below the $P_{migrate}$, the task is migrated from M_{max} to M_{min}. The loads on the nodes are updated, and the process is repeated until no further load balance is possible. This probability-based load balancing ensures optimal task-to-node allocation, minimising local imbalances.

4.3. Weight-based reallocation of tasks

While probability-based load balancing evens out the load, it does not consider node processing speeds during the migration of the task. To address this, we perform weight-based task reallocation. In this step, the tasks are reallocated by calculating the weight for each node. The tasks are migrated from overloaded nodes based on the probability PW, as shown in Eq. (8).

$$PW = 1 - W_j \tag{8}$$

Let Wj denote the weight of node j for task assignment, computed as in Eq. (9).

$$W_j = \alpha \cdot \frac{TL_j}{TL_{max}} - \beta \cdot \frac{\mu_j}{\mu_{max}} \tag{9}$$

A higher value of W_j indicates that the node is given a higher priority for receiving the task.
TL_j denotes the load on node M_j, and a higher value of TL_j means the node is overloaded.
TL_{max} denotes the maximum load among all nodes, and μ_{max} denotes the maximum speed among all the nodes.
α and β are control parameters Indicating the priorities for resource utilization and total execution time. The values are chosen between 0 to 1. In the study, the values are set equal, indicating that both parameters are equally important.

5. Optimization Model for Load Balancing

To ensure system stability, there are different performance metrics measured in load balancing [36]. Some of the metrics are makespan, response time, resource utilization, communication time, throughput, etc. The makespan and resource utilization are the metrics measured in the study. As a result of good load balancing, the throughput and fair distribution are also measured.
Makespan:
Makespan is the amount of time required to complete all the tasks in the system. It is the time elapsed between the start of the first task and the end of the last task. The objective is to minimize the makespan (MS), maintaining the load balance in the system. The MS is given in Eq. (10)

$$MS_i = \max_i ECT_i \; \forall i = 1, 2, 3, ..., m \tag{10}$$

The average makespan, AvgMS, as calculated in Eq. (11)

$$AvgMS = \frac{1}{m}\sum_{i=1}^{m} MSi \tag{11}$$

The objective F_1 is to minimize AvgMS.

F_1 = minimize AvgMS(TL_1, TL_2, ….., TL_m)

Resource Utilization:
Resource utilization is a measure of how much the available resources are currently being utilized. Load balancing should ensure that resources are utilized proportionally to their capabilities, and the objective function is to maximize resource utilization.
Let RU_i denote the resource utilization of node 'i as given in Eq. (12)

$$RU_i = \frac{TL_i}{\mu_i} \; \forall \; i = 1, 2, 3, \ldots, m \tag{12}$$

To measure the average utilization of all nodes in the system is given in Eq. (13)

$$AvgRU = \frac{1}{m}\sum_{i=1}^{m} RUi \tag{13}$$

The objective is to maximize the average utilization of the system, and the objective function can be stated as in (14)

F_2=MaximizeAvgRU (TL_1,TL_2,…..,TL_m) (14)

Load Balancing as a Maximization Problem:
Combining both objectives, load balancing can be described as a maximization problem and is expressed as in Eq. (15)

Maximize F = F_2–F_1 (15)

In this study, PSO is employed to meet the objectives of load balancing such that each node gets optimal loads L_1^*, L_2^*,…, L_m^* to achieve a steady state of the system. The fitness function is calculated as in Eq. (15)

5.1. Particle swarm optimization

Optimization is a technique that involves finding minima or maxima of some function in a feasible region. The steps in the PSO
The algorithm includes initializing the population with a specified setting of initial positions and velocities for each particle, and iteratively updating these positions and velocities.
Initial Population: In the study initial population is generated randomly. Each particle in the population represents the candidate solution, indicating random allocation of tasks to nodes.
Initial Position: Each particle's position vector, represented by x_i, is initialised randomly as in Eq. (16).

$x_i = [\, x_{i1}, x_{i2}, \ldots . x_{in}]$ (16)

Where $x_{i,j} \in \{1, 2, \ldots . m\}$
Initial Velocity: This component is used to move the position of each particle in the search space for the optimal solution.
Fitness function: In each iteration, the objective function is evaluated according to Eq. (15) to meet the objectives defined for the problem.
Updation: During each iteration, the position and velocity of each particle are adjusted based on their own experience, P_{best}, and the influence of their neighbours, G_{best}. The position and velocity are updated according to Eq. (17) and Eq. (18). The position value represents the allocation of a task to a node, and the value is rounded to lie between 1 to m.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (17)$$

$$v_i(t+1) = \omega v_i(t) + c_1 r_1 (P_{best} - x_i(t)) + c_2 r_2 (G_{best} - x_i(t)) \quad (18)$$

Where i represents particle index; c1 and c2 are acceleration coefficients; $x_i(t)$ and $v_i(t)$ are particle position and velocity at time t respectively, inertia weight is indicated by ω; r1 and r2 denote randomly chosen values; P_{best} refers to a particle's personal best solution, while G_{best} represents the best solution achieved by the entire swarm at time t.
In PW-PSO, the fitness function F in Eq. (15) aggregates the two main objectives: maximizing Average Resource Utilization (AvgRU) while minimizing Average Execution Completion Time (AvgECT). A higher value of F, therefore, corresponds to a task allocation that uses the heterogeneous nodes more evenly and finishes tasks more quickly, in line with the probability-based migration and weight-based reallocation rules, producing an updated allocation vector $x_i(t)$ and $v_i(t)$ denoting the position and velocity. For each particle, the fitness F_i of its current position task to node assignment is compared with its personal best and the global best. If F_i is larger, the corresponding P_{best} and G_{best} are updated. The standard PSO update in Eq. (18) then maps particles towards the allocations with a higher F value. In this way, the swarm's movement is explicitly guided toward configurations that best satisfy the combined objectives encoded in the fitness function, highly utilized, well-balanced nodes with reduced completion time.
The step-by-step procedure for the hybrid methodology is given in Algorithm 1.

Algorithm 1 Proposed method PW-PSO

Input: Initialize particle swarm with random allocation of tasks to nodes
Output: Optimized Task to Resource Allocation
Initialize velocities, P_{best}, and G_{best}
for $i \leftarrow 1$ to 20 **do**
  Update velocity and position of each particle
  Compute AvgRU $\leftarrow \frac{1}{m}\sum_{i=1}^{m} \text{RU}_i$
  Compute AvgECT $\leftarrow \sum_{i=1}^{m} \frac{\text{TL}_i}{\mu_i}$
  Compute Fitness: $F \leftarrow$ AvgRU − AvgECT
  Compute Load and Identify Overloaded node $M_{\max}$ and Underloaded node $M_{\min}$
  for $j \leftarrow 1$ to n **do**
    Compute Probability based migration $P_{\text{migrate}} \leftarrow \frac{\text{TL}_{\max} - \text{TL}_{\min}}{\text{TL}_{\max}}$
    With probability P_{migrate}, move task to $M_{\min}$
    Compute node's weight using it's load and processing speed
    $$W_j \leftarrow \alpha \cdot \frac{\text{TL}_j}{\text{TL}_{\max}} - \beta \cdot \frac{\mu_j}{\mu_{\max}}$$
    Reallocate tasks based on weights to achieve load balancing
  end for
  Update P_{best}: $x_i(t+1) \leftarrow x_i(t) + v_i(t+1)$
  Update G_{best}: $v_i(t+1) \leftarrow \omega v_i(t) + c_1 r_1 (P_{\text{best}} - x_i(t)) - c_2 r_2 (G_{\text{best}} - x_i(t))$
end for

6. Simulation Setup

The implementation of load balancing using PSO involves setting a few PSO parameters to control the algorithm's behaviour and optimize the problem. These parameters include problem dimensionality, swarm size, particle velocity, cognitive learning rate, inertia weight, random factors, and social learning rate. Initially, the population is generated by randomly assigning tasks to nodes. The simulations are conducted using the CloudSim simulator. The simulation parameter settings are listed in Table II.
Scenario 1: The simulations are conducted with the nodes set at 200 and the task size varying from 1000 to 5000 in steps of 1000.
Scenario 2: The simulations are conducted with a constant task size of 5000, while the number of nodes varies from 40 to 200 in steps of 40.

7. Results and Discussions

During peak usage, some servers receive excessive requests and become overloaded, causing request failures. Hence, in such situations, load balancing becomes crucial to improve the system's stability. The key findings in the proposed study are the total execution time and resource utilization. As a result of load balancing, throughput and fairness of load distributions are measured as in Eq. (19) and Eq. (20), respectively.
Throughput(T): Throughput is the number of tasks completed per unit time, as in Eq. (19). It is used to evaluate the efficiency of a distributed system.

$$T = \frac{\text{Total tasks completed}}{\text{Total time taken}} \tag{19}$$

Fairness Index(FI): In dynamic load balancing, fairness is a measure of how evenly tasks are distributed across all the available computational loads. The Jain's Fairness Index (JFI) is calculated as shown in Eq. (20).

$$JFI = \frac{(\sum_{i=1}^{m} RU_i)^2}{m.(\sum_{i=1}^{m} RU_i^2)} \tag{20}$$

Table 2: Simulation Parameters

Type	Parameters	Values
Node characteristics	Number of Nodes	40-200
	Node memory capacity	128MB,256MB, 512MB
	Node speed	500 – 1000 MIPS
Swarm parameters	Population size	30
	Lower bound of x_0	1
	Upper bound of x_0	40-200
	Lower bound on v_0	-0.5
	Upper bound on v_0	0.5
	Number of iterations	20
	C1	1.9
	C2	1.9
	Inertia weight ω	Constant – 0.9
Task characteristics	Number of tasks	1000-5000
	Type of task	Independent Atomic Non-preemptive
Hyper parameters	α	.5
	β	0.5
	t_{max}	0.8
	t_{min}	0.2

Tables 3-6 show the comparative results for makespan, resource utilization, throughput, and fairness index, respectively, for Scenario 1.
Table 3 compares the makespan of the proposed PW-PSO method with other techniques examined in the study. The row values represent the makespan in seconds for a varying number of tasks. Fig. 3 shows the comparison of the makespan measured for existing techniques and the proposed study, clearly demonstrating that the proposed methodology achieves a better makespan, emphasizing its effectiveness. The dynamic task allocation by the algorithm effectively prevents slower nodes from increasing execution time, resulting in consistently low makespan values across various task loads. This enhances the robustness and efficiency of the proposed algorithm in maintaining optimal performance under different workload conditions.

Table 3: Comparison of Makespan for Varying Task Sizes

No. of Tasks	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW-PSO (Proposed)
1000	103	110	96	88	81
2000	202	214	185	162	154
3000	310	335	284	256	225
4000	389	411	364	326	297
5000	477	494	446	402	368

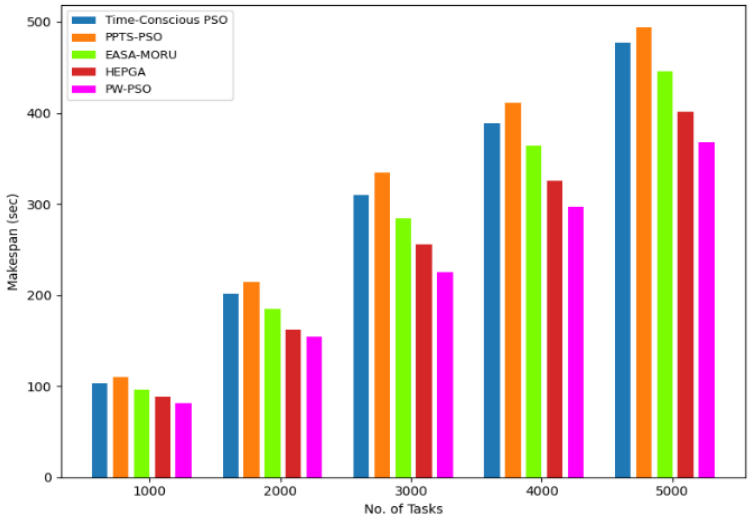


Fig. 3: Comparison of Makespan for Varying Task Sizes.

Table 4 shows the resource utilization of various existing methods and the proposed method. The values in the table represent the resource utilization of the computational nodes. Fig. 4 depicts the comparison of resource utilization measured for the existing methods in the study and the proposed methodology for Scenario 1. As shown in Fig. 4, the proposed method outperforms other existing methods significantly. The probabilistic task allocation introduces controlled randomness, preventing premature convergence and ensuring optimal resource utilization outcomes. Moreover, the weighted load migration demonstrates dynamic adaptability by efficiently redistributing tasks to lightly loaded nodes without compromising system performance. With an increasing number of tasks, the proposed algorithm shows good performance, demonstrating robustness and scalability across diverse computational scenarios.

Table 4: Comparison of Resource Utilization for Varying Task Sizes

No. of Tasks	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW-PSO (Proposed)
1000	0.90	0.89	0.85	0.89	0.93
2000	0.91	0.89	0.86	0.90	0.94
3000	0.92	0.90	0.87	0.91	0.94
4000	0.93	0.91	0.88	0.91	0.95
5000	0.94	0.92	0.89	0.92	0.95

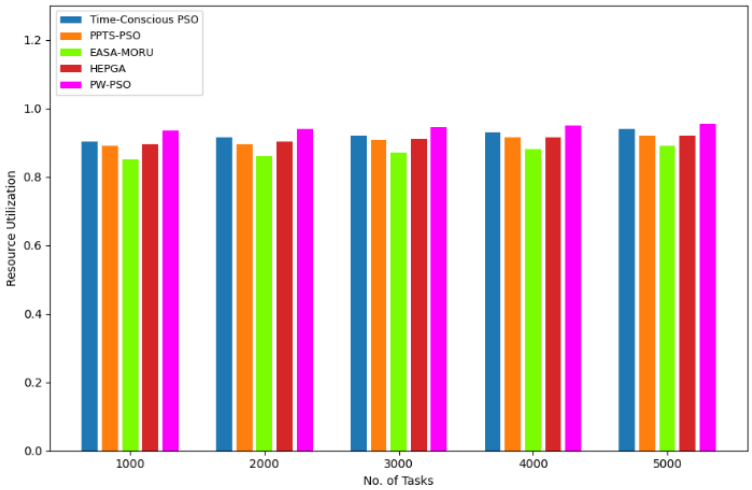


Fig. 4: Comparison of Resource Utilization for Varying Task Sizes.

System efficiency is evaluated using throughput and fairness index, with the proposed method outperforming the existing techniques. Table 5 presents the throughput of the proposed study in comparison with existing approaches. The values in the table indicate the number of tasks completed per unit of time for each of the methods. Fig. 5 depicts the comparative analysis of throughput between existing methods and the proposed method. The figure clearly indicates that the proposed method achieves higher throughput than the other techniques. The adaptive task allocation employed by the proposed algorithm enhances parallel task execution, effectively reducing the makespan and thereby increasing throughput. Furthermore, the load migration model continually monitors the node load conditions and transfers tasks dynamically to the most suitable nodes, enabling the successful completion of a greater number of tasks within the distributed system.

Table 5: Comparison of Throughput for Varying Task Sizes

No. of Tasks	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW-PSO (Proposed)
1000	11.8	11.48	12.2	11.2	12.46
2000	12.3	11.65	12.6	11.5	13.1
3000	12.6	11.89	13,	11.8	13.4
4000	12.94	12.16	13.2,	12.2	13.52
5000	13.3	12.9	13.4	12.5	13.6

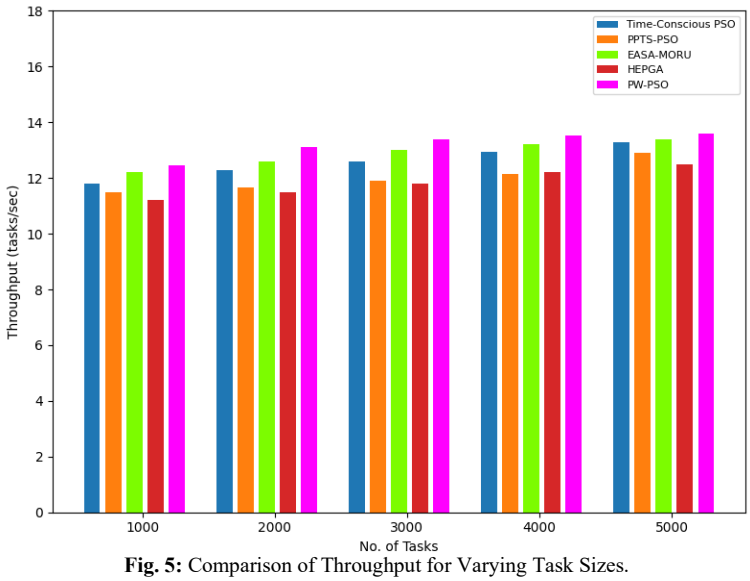


Fig. 5: Comparison of Throughput for Varying Task Sizes.

Table 6 presents the fairness index, which reflects the distribution of tasks across computational nodes. The values range from 0 to 1, with values closer to 1 indicating more equitable task allocation. Fig. 6 illustrates the fairness in task distribution among the evaluated methods, with the PW-PSO algorithm attaining the highest fairness index. The improved fairness is attributed to the algorithm's probabilistic redistribution and load migration mechanisms, which dynamically monitor and adjust load imbalances. Consequently, the proposed algorithm achieves equitable workload distribution across all nodes and mitigates task starvation on previously overloaded nodes.

Table 6: Comparison of Fairness Index

No. of Tasks	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW- PSO (Proposed)
1000	0.92	0.91	0.947	0.89	0.965
2000	0.927	0.915	0.95	0.896	0.97
3000	0.932	0.923	0.955	0.905,	0.976
4000	0.936	0.93	0.961	0.916	0.979
5000	0.941	0.934	0.968	0.925	0.983

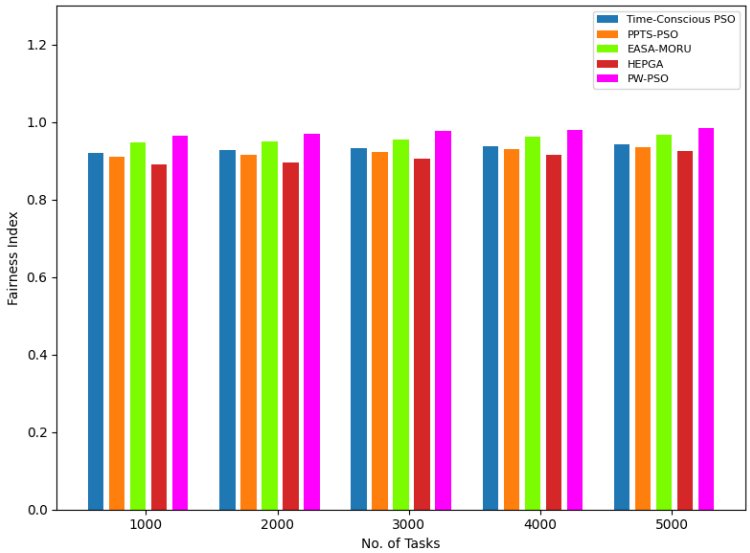


Fig. 6: Comparison of Fairness Index for Varying Task Sizes.

The proposed method is also tested for its adaptability to a varying number of computational nodes.
Tables 7-10 present the performance evaluation for Scenario 2, where the number of tasks is fixed at 5,000, and the number of nodes ranges from 40 to 200 in increments of 40. In each table, the column values indicate the results of the performance metric for different methods and the proposed method.
Table 7 summarizes the makespan obtained for 5000 tasks under varying numbers of nodes. Each column value represents the task completion time of various approaches. Fig. 7 illustrates the comparative makespan analysis of the proposed method against existing techniques. The proposed algorithm consistently yields superior results, highlighting its adaptability to larger networks with increased nodes, which provide more options for effective load migration. The probability-based approach selectively migrates tasks based on the calculated migration probability, effectively reducing task queue lengths. Additionally, the weighted-based method strategically redistributes tasks according to the computational capabilities of nodes, further enhancing the system's overall efficiency and performance.

Table 7: Comparison of Makespan for Varying Nodes

No. of nodes	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW- PSO (Proposed)
40	1988	2132	1911	1823	1784
80	995	1065	961	923	898
120	785	823	723	688	604
160	652	701	598	523	459
200	477	494	446	402	368

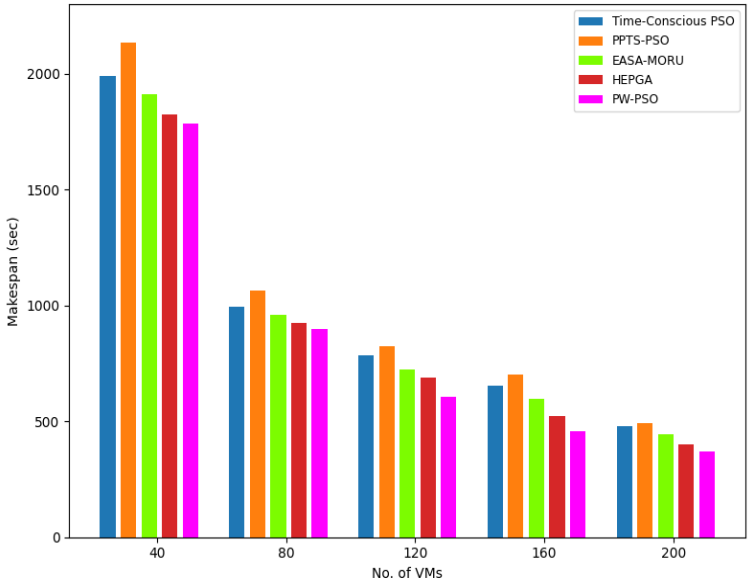


Fig. 7: Comparison of Makespan for Varying Numbers of Nodes.

Table 8 reports the resource utilization values for varying numbers of nodes, while Fig. 8 compares the performance of the proposed and existing methods. The results indicate that the proposed algorithm achieves higher resource utilization. As the number of nodes increases, flexibility in selecting nodes for task migration improves. The weighted method further allocates tasks in proportion to node capabilities. Consequently, the adaptive strategy attains efficient task allocation, leading to improved average resource utilization across all nodes.

Table 8: Comparison of Resource Utilization

No. of nodes	Time-Conscious PSO	PPTS-PSO	EASA- MORU	HEPGA	PW- PSO (Proposed)
40	0.962	0.943	0.915	0.94	0.978
80	0.958	0.938	0.91	0.935	0.972
120	0.952	0.931	0.905	0.93	0.966
160	0.946	0.925	0.898	0.924	0.96
200	0.94	0.92	0.89	0.92	0.955

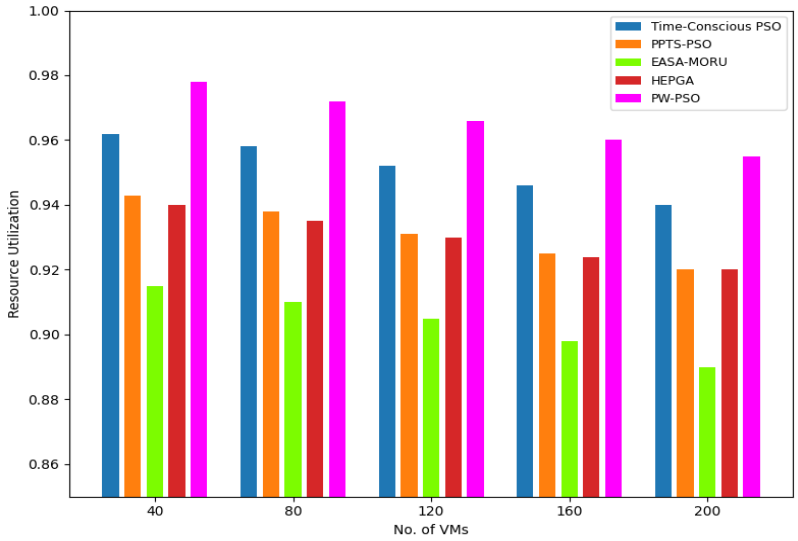


Fig. 8: Comparison of Resource Utilization for Varying Numbers of Nodes.

In Table 9, the values represent the throughput obtained for the existing methods and the proposed method, while Fig. 9 shows a comparative analysis of throughput for varying numbers of nodes. The graphical results demonstrate that the proposed method achieves superior throughput due to its dynamic capability to effectively balance the workload across a larger and more diverse set of nodes. The increased number of nodes enhances parallel task execution, thereby significantly improving the overall system throughput and performance.

Table 9: Comparison of Throughput for Varying Number of Nodes

No. of nodes	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW- PSO (Proposed)
40	2.3	2.0	2.7	1.7	2.9
80	4.5	4.1	5.1	3.7	5.6
120	7.2	6.7	7.7	6.1	8.3
160	9.8	9.3	10.3	8.9	11.0
200	12.0	11.2	12.8	10.8	13.6

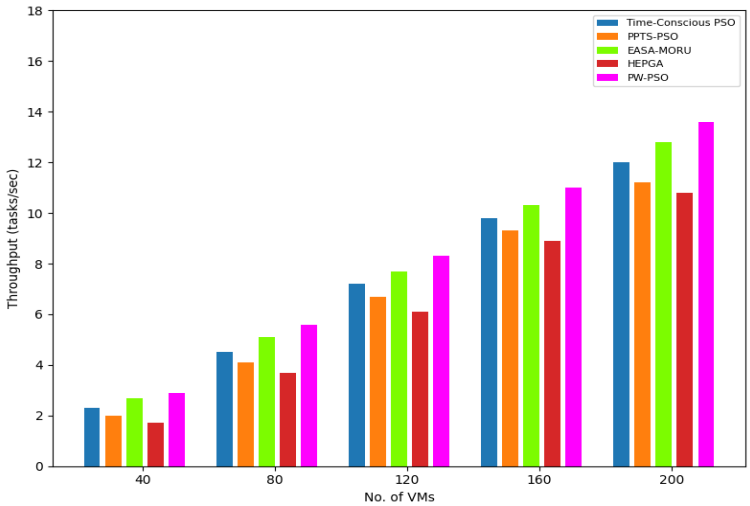


Fig. 9: Comparison of Throughput with Varying Numbers of Nodes.

Table 10 presents the fairness index for 5000 tasks under varying node numbers, while Fig. 10 compares the task distribution fairness between the proposed algorithm and existing methods. The figure demonstrates that as the number of nodes increases, the proposed method consistently improves the Jain's Fairness Index (JFI) due to its adaptive capability to consider both current load conditions and node processing speeds. Furthermore, the algorithm's flexibility in balancing loads improves with the addition of nodes, facilitating a more uniform distribution of tasks across the nodes and enhancing overall system fairness and performance.

Table 10: Comparison of Fairness Index for Varying Number of Nodes

No. of nodes	Time-Conscious PSO	PPTS-PSO	EASA-MORU	HEPGA	PW- PSO (Proposed)
40	0.927	0.917	0.951	0.908	0.971
80	0.930	0.922	0.957	0.913	0.975
120	0.935	0.926	0.96	0.917	0.978
160	0.938	0.931	0.965	0.920	0.981
200	0.941	0.934	0.968	0.925	0.983

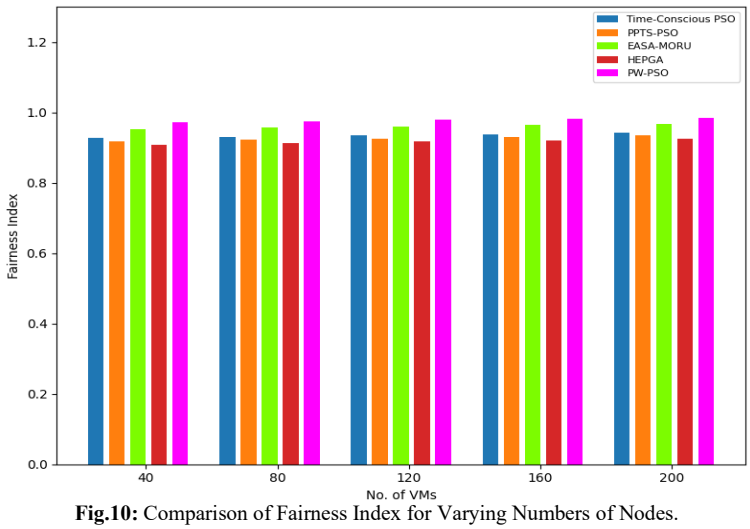


Fig.10: Comparison of Fairness Index for Varying Numbers of Nodes.

The proposed PW-PSO method outperforms the baseline algorithms considered in this study, consistently delivering superior results across multiple performance metrics. The probabilistic approach of scheduling and weighted load reallocation continuously monitors and balances the load. The work, combined with PSO, highlights a novel method to achieve the objective of optimal allocation of tasks to nodes.

8. Conclusion

The integrated Probability-Weighted load balancing approach using Particle Swarm Optimization (PW-PSO) significantly enhances the efficiency of load distribution in heterogeneous distributed computing environments. By enabling continuous monitoring and dynamic redistribution of workloads, the proposed method ensures optimal task allocation across nodes. It effectively assigns tasks in proportion to

the computational capabilities of each node, resulting in a fair and balanced distribution of load. This leads to improvements in system performance with reduced makespan, increased resource utilization, higher throughput, and an improved fairness index. Simulation results confirm that the proposed PW-PSO algorithm consistently outperforms existing approaches such as Time-Conscious PSO, PPTS-PSO, EASAMORU, and HEPGA, demonstrating its potential for more efficient and scalable task scheduling. Furthermore, future research can explore the fine-tuning of PSO control parameters and hyperparameters to further optimize performance.

References

[1] Handur Vidya S., Santosh L. Deshpande, and Prakash R. Marakumbi. "Particle Swarm Optimization for Load Balancing in Distributed", Turkish Journal of Computer and Mathematics Education (TURCOMAT),Vol.12 No.1S,257-265,04,(2021). https://doi.org/10.17762/turcomat.v12i1S.1766

[2] R. Tian, X. Chen, T. Song & T. Wang, "Cost Optimization of Queueing Systems with Flexible Priorities and Heterogeneous Servers," Engineering Letters, vol. 33, no. 6, EL_33_6_34, (2025), (Accessed on June 26 2025).

[3] Elmagzoub M., Syed Darakhshan, Shaikh Asadullah ,Islam Noman, Alghamdi Abdullah,and Rizwan Syed." A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment", Electronics. November,(2021) (Accessed on June,08,2025). https://doi.org/10.3390/electronics10212718.

[4] T. Islam and M. S. Hasan,(2018)" A performance comparison of load balancing algorithms for cloud computing", International Conference on the Frontiers and Advances in Data Science (FADS),pp. 130 - 135, (2018) https://doi.org/10.1109/FADS.2017.8253211.

[5] H. Rahmawan and Y. S. Gondokaryono,"The simulation of static load balancing algorithms", International Conference on Electrical Engineering and Informatics, Bangi, Malaysia, pp 640-645, (2009) https://doi.org/10.1109/ICEEI.2009.5254739.

[6] Elmagzoub M., Syed Darakhshan, Shaikh Asadullah, Islam Noman, Alghamdi Abdullah,and Rizwan Syed." A Survey of Swarm Intelligence Based Load Balancing Techniques in Cloud Computing Environment", Electronics. November,(2021) https://doi.org/10.3390/electronics10212718. (Accessed on June,08,2025).

[7] Simone A. Ludwig, Azin Moallem,"Swarm Intelligence Approaches for Distributed Load Balancing on the Grid", Journal of Grid Computing, (2011). https://doi.org/10.1007/s10723-011-9180-5.

[8] Vidya S. Handur and R. Marakumbi Prakash," Response time analysis of dynamic load balancing algorithms in Cloud Computing", IEEE Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4),(2020). https://doi.org/10.1109/WorldS450073.2020.9210305.

[9] Priyavada and Binay Kumar," A Review of Modified Particle Swarm Optimization Method, Soft Computing and Optimization", SCOT, Springer Proceedings in Mathematics & Statistics, vol 404. Springer, Singapore,(2021) https://doi.org/10.1007/978-981-19-6406-0_3.

[10] Yo-Bo Wang, Jei-Sheng Wang, Xiao-Fei Sui," Improved Particle Swarm Optimization Algorithm with Logistic Function and Trigonometric Function for Path Planning Problems", Engineering Letters, vol 33, no 2, EL_33_2_22,(2025), (Accessed on June 24,2025).

[11] Ning Qin & Xuelei Meng, "High-speed Train Rescheduling Based on a New Kind of Particle Optimization Algorithm," Engineering Letters, vol. 31, no. 2, pp 640–647, (2023), (Accessed on June 20,2025).

[12] Handur Vidya S., and Santosh L. Deshpande," Artificial Bee Colony Optimization-Based Load Balancing in Distributed Computing Systems—A Survey", Smart Trends in Computing and Communications, Springer, Singapore,733-740,(2022). https://doi.org/10.1007/978-981-16-9967-2_69.

[13] Huyin Zhang and Kan Wang, "Research of dynamic load balancing based on stimulated annealing algorithm", International Journal of Embedded Systems, Vol. 10, No. 3, pp 188–195, (2018), https://doi.org/10.1504/IJES.2018.091777. (Accessed on June 02,2025).

[14] Handur Vidya S, Supriya Belkar, Santosh L. Deshpande, Prakash R.Marakumbi,"Study of load balancing algorithms for Cloud Computing", Second IEEE International Conference on Green Computing and Internet of Things (ICGCIoT),(2018). https://doi.org/10.1109/ICGCIoT.2018.8753091.

[15] Hanamakkanavar Amit S., and Vidya S. Handur,"Load balancing in distributed systems: a survey",International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE) ISSN: 0976-1353, Volume 14, Issue, (2015).

[16] Md Oqail Ahmad and Rafiqul Zaman Khan," An efficient load balancing scheduling strategy for cloud computing based on a hybrid approach", International Journal of Cloud Computing, Vol. 9, No. 4, pp 453–469,(2020), https://doi.org/10.1504/IJCC.2020.112317. (Accessed on June 21, 2025)

[17] Essam H. Houssein, Ahmed G. Gad, Yaser M. Wazery, Ponnuthurai Nagaratnam Suganthan," Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends", Swarm and Evolutionary Computation,(2021), https://doi.org/10.1016/j.swevo.2021.100841, (Accessed on June 01,2025).

[18] Brototi Mondal, "Load balancing in cloud computing using cuckoo search algorithm", International Journal of Cloud Computing, Vol. 13, No. 3, pp 267–284,(2024), https://doi.org/10.1504/IJCC.2024.139594. (Accessed on June 02,2025).

[19] Subhadarshini Mohanty, Prashanta Kumar Patra, Mitrabinda Ray, Subasish Mohapatra," A Novel Meta-Heuristic Approach for Load Balancing in Cloud Computing", International Journal of Knowledge-Based Organizations, Volume 8, Issue 1, January-March (2018), https://doi.org/10.4018/IJKBO.2018010103. (Accessed on June, 06,2025).

[20] Vahid A C, Seyed Naser Razavi," Resource Allocation in Cloud Environment Using Approaches Based Particle Swarm Optimization", International Journal of Computer Application Technology and Research, Volume -6, Issue 2, 87-90, (2017), https://doi.org/10.7753/IJCATR0602.1003, (Accessed on :May,10,2025).

[21] Md Oqail Ahmad and Rafiqul Zaman Khan," An efficient load balancing scheduling strategy for cloud computing based on a hybrid approach", International Journal of Cloud Computing, Vol. 9, No. 4, pp 453–469,(2020), https://doi.org/10.1504/IJCC.2020.112317 .(Accessed on June 21, 2025)

[22] 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,January 2019, https://doi.org/10.3103/S0146411619010024. (Accessed on May,20,2025).

[23] Fatemeh Ebadifard, Seyed Morteza Babamir, "A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment", Concurrency and Computation Practice and Experience, Volume 30, Issue 12, (2017), https://doi.org/10.1002/cpe.4368, (Accessed on May 12,2025).

[24] D. Komalavalli and T. Padma, "Swarm intelligence-based task scheduling algorithm for load balancing in cloud system", International Journal of Enterprise Network Management, Vol. 12, No. 1, pp 1–16, (2021) https://doi.org/10.1504/IJENM.2021.112669. (Accessed on June 21,2025).

[25] Nabi, S., Ahmad, M, Ibrahim, M., Hamam, H," AdPSO: Adaptive PSO-Based Task Scheduling Approach for Cloud Computing", Sensors, 22, 920, (2022), https://doi.org/10.3390/s22030920. (Accessed on May 21 2025)

[26] Purshottam J. Assudani and P. Balakrishnan, "A novel bio-inspired approach for VM load balancing and efficient resource management in cloud", International Journal of Ad Hoc and Ubiquitous Computing, Vol. 40, No. 1-3, pp 214–224,(2022), https://doi.org/10.1504/IJAHUC.2022.123541. (Accessed on June 21,2025).

[27] Arabinda Pradhan, Sukant Kishoro Bisoy," A novel load balancing technique for cloud computing platform based on PSO, Journal of King Saud University – Computer and Information Sciences,34, 3988–3995,(2022) https://doi.org/10.1016/j.jksuci.2020.10.016, (Accessed on June 10,2025).

[28] Ahmad M. Manasrah, 'Dynamic weighted VM load balancing for cloud-analyst", International Journal of Information and Computer Security, Vol. 9, No. 1-2, pp 5–19,(2017), https://doi.org/10.1504/IJICS.2017.10003596, (Accessed on June 20,2025).

[29] Arabinda Pradhan, Sukant Kishoro Bisoy," A novel load balancing technique for cloud computing platform based on PSO, Journal of King Saud University – Computer and Information Sciences,34, 3988–3995,(2022), https://doi.org/10.1016/j.jksuci.2020.10.016. (Accessed on June 10,2025).

[30] M. Menaka, K.S. Sendhil Kumar, "Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing", International Journal of Cognitive Computing in Engineering, Volume 5, Pages 192-198, ISSN 2666-3074,(2024), https://doi.org/10.1016/j.ijcce.2024.05.002

[31] Hind Mikram, Said El Kafhali, Youssef Saadi, "HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment", Simulation Modelling Practice and Theory, Volume 130, (2024) https://doi.org/10.1016/j.simpat.2023.102864. (Accessed on June, 03,2025)

[32] Talha, Adnane, & Malki, Mohammed,"PPTS-PSO: a new hybrid scheduling algorithm for scientific workflow in a cloud environment", Multimedia Tools and Applications.Vol 82, No.21,(2023) https://doi.org/10.1007/s11042-023-14739-w. (Accessed on June 04,2025).

[33] K. Karim F, Ghorashi S, Alkhalaf S, H. A. Hamza S, Ben Ishak A, Abdel-Khalek S," Optimizing makespan and resource utilization in cloud computing environment via evolutionary scheduling approach", PLoS ONE 19(11): e0311814, (2024), https://doi.org/10.1371/journal.pone.0311814. (Accessed on June 02 2025).

[34] Hicham BEN ALLA, Said BEN ALLA and Abdellah EZZATI DP-"ARTS: Dynamic Prioritization for Adaptive Resource Allocation and Task Scheduling in Cloud Computing", IAENG International Journal of Computer Science, Volume 52, Issue 3, Pages 793-807, (2025), https://doi.org/10.1109/UNet62310.2024.10794725. (Accessed on June 01,2025).

[35] Soha Rawas and Ahmed Zekri, "EEBA: Energy-Efficient and Bandwidth Aware Workload Allocation Method for Data-intensive Applications in Cloud Data Centers", IAENG International Journal of Computer Science, Volume 48, Issue 3: (2021) (Accessed on June 03,2025).

[36] Ali M. Alakeel," A Guide to Dynamic Load Balancing in Distributed Computer Systems", IJCSNS International Journal of Computer Science and Network Security, Vol. 10, No.6, (2010) (Accessed on June,12,2025).