International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/36 Volume 12, Issue 4, July – August (2025)

REVIEW ARTICLE

A Comprehensive Evaluation of Swarm Intelligence Metaheuristics for Efficient Load Balancing in Cloud Computing

Vaishali Mehta
Department of Computer Science and Applications, Panjab University, Chandigarh, India.
✉ vaishali.m@pu.ac.in

Anu Gupta
Department of Computer Science and Applications, Panjab University, Chandigarh, India.
anugupta@pu.ac.in

Received: 15 May 2025 / Revised: 20 July 2025 / Accepted: 06 August 2025 / Published: 30 August 2025

Abstract – Cloud computing has transformed resource provisioning across distributed infrastructures, introducing both opportunities and challenges for efficient resource management. A primary challenge in resource management is load balancing, which seeks to optimize overall system performance by intelligently mapping user workloads to available computing resources. Due to the combinatorial complexity in dynamic cloud environments, heuristic and metaheuristic approaches have become essential. Among these, nature-inspired metaheuristic algorithms, which are modeled on collective behaviors observed in nature, have demonstrated strong potential for addressing load-balancing problems. This paper provides a detailed review and performance comparison of popular nature-inspired swarm intelligence algorithms, including Ant Colony Optimization, Particle Swarm Optimization, the Firefly Algorithm, and the Bat Algorithm. Each algorithm was implemented and evaluated within the CloudSim simulation framework under varying cloudlet workloads. Performance was evaluated using key performance metrics related to load-distribution efficiency. The analysis identifies the strengths and limitations of each algorithm, providing evidence-based insights into their suitability for various cloud computing scenarios. Results demonstrate that nature-inspired algorithms hold considerable promise for enhancing load-balancing efficiency of cloud computing environments.

Index Terms – Load Balancing, Cloud Computing, Resource Management, Metaheuristic Algorithms, Bio-Inspired, Swarm Intelligence, Particle Swarm Optimization.

1. INTRODUCTION

Cloud computing underpins contemporary digital services by delivering scalable and flexible access to processing, storage, and networking resources through pay-as-you-go models. Its dynamic nature often results in request overload across critical components like devices, storage systems, and data centers [1]. According to IDC, the worldwide public cloud services sector is forecasted to grow to USD 1 trillion by 2026, driven by the adoption of AI, edge computing, and data-intensive applications [2]. This growth has led to a rapid proliferation of cloud-based applications, intensifying the demand for efficient and dynamic resource management across distributed infrastructures. Cloud environments must accommodate fluctuating workloads, user demands, and application heterogeneity, all while maintaining service quality and minimizing operational costs. However, the variable nature of resource requests can often lead to uneven distribution of workloads across cloud infrastructure, creating bottlenecks in devices, data centers, and storage systems [3].

Thus, efficient resource management sits at the heart of cloud economics and sustainability. Among its sub-domains, Load Balancing (LB) is pivotal, which ensures fair task distribution among Virtual Machines (VMs), preventing overload or underutilization and maintaining uniform performance across data centers. Effective LB not only improves overall system performance but also enhances resource utilization, reduces energy consumption, and ensures compliance with Service-Level Agreements (SLAs). Since cloud applications exhibit varying task arrival patterns, diverse execution times, and evolving QoS constraints, traditional deterministic approaches often fall short in achieving real-time, optimal scheduling solutions [4]. The LB problem, inherently combinatorial and NP-hard, becomes even more complex in heterogeneous environments involving thousands of tasks and hundreds of VMs. To tackle this complexity, researchers have utilized many heuristic as well as metaheuristic algorithms [5]. The heuristic methods are often used to solve a specific problem, yielding faster results but lacking generalizability. They tend to get trapped in local optima and are unable to adapt to changes in system dynamics or constraints. Thus, researchers

REVIEW ARTICLE

are increasingly turning to metaheuristic techniques, including Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Firefly Algorithm (FA), and Genetic Algorithms (GA), which are designed to explore vast solution spaces through intelligent iteration and stochastic decision-making. These methods balance the process of exploration (investigating new regions of the solution domain) with exploitation (enhancing solutions that have proven effective), making them suitable for the dynamic cloud environments where workloads and requirements are continuously evolving [6].

Among metaheuristics, nature-inspired algorithms have attracted significant attention. These are generally classified into swarm-intelligence and non-swarm-intelligence methods. Swarm intelligence (SI) algorithms are modeled after the cooperative behavior observed in social organisms such as ants, bees, birds, or fish, and include approaches like ACO, PSO, FA, and BAT algorithms. Non-swarm methods, on the other hand, derive inspiration from evolutionary biology (e.g., Genetic Algorithms) or physical processes (e.g., Simulated Annealing) [7]. Swarm-intelligence techniques offer multiple advantages for cloud load balancing: they are inherently parallel, easy to implement, require fewer parameters, and adapt efficiently to dynamic changes in task arrivals or resource availability. The flexibility, scalability, and modest parameter requirements of these algorithms make them attractive for cloud LB, inspiring a surge of empirical studies and hybrid approaches over the last decade [4].

Existing reviews and taxonomic studies of metaheuristic scheduling and LB techniques in cloud computing offer valuable overviews, yet they seldom capture swarm-intelligence algorithms across the full range of characteristics such as objective functions, dynamic adaptation capabilities, and energy-aware mechanisms, leaving an important dimension underexplored. The performance evidence remains fragmented because comparisons are run on diverse simulators, metrics, and workload profiles, which complicate cross-study synthesis, and the discussion typically stops short of translating algorithmic insights into actionable guidance for practitioners who must deploy and tune these methods on real-world cloud platforms [8].

Given this landscape, a comprehensive, reproducible, and performance-driven review is needed, one that not only categorizes algorithms but also evaluates them under a unified framework. This study addresses that need by providing a thorough and systematic analysis, aiming to enhance operational comprehension of four canonical swarm-intelligence LB algorithms, including Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and BAT Optimization Algorithm, implemented under a uniform CloudSim environment. The paper examines the unique traits, benefits, and drawbacks of each algorithm, and these algorithms are benchmarked across a set of standard LB performance metrics, which include makespan, response time, degree of imbalance, resource utilization, standard deviation, and processing speed, providing a decision aid for system architects. Table 1 presents a comparative overview of this study with prior research efforts.

Table 1 Comparison of Present Research vs Previous Research

Ref No.	State of the art	Taxonomy	Comparative Analysis	Graphical Analysis
[1]	No	Yes	Yes	Yes
[4]	Yes	No	Yes	No
[6]	Yes	No	Yes	No
[7]	Yes	No	Yes	No
[8]	Yes	No	Yes	No
Present Research	Yes	Yes	Yes	Yes

The following are the contributions of this study:

- It presents an enhanced taxonomy of SI algorithms for implementing load balancing in a cloud-based environment.
- It conducts an empirical comparison of algorithm performance using a consistent simulation platform and workload datasets.
- It provides insights into algorithm suitability, trade-offs, and use-case alignment, serving as a decision-making aid for researchers and practitioners.

In the rest of this paper, Section 2 outlines the foundational concepts of cloud computing, resource management, and load balancing. A detailed literature review of swarm intelligence metaheuristic algorithms developed for load balancing is provided in Section 3. Further, Section 4 conducts a

REVIEW ARTICLE

comprehensive performance evaluation of the selected algorithms. Section 5 discusses challenges and outlines directions for future work, and lastly, Section 6 presents the conclusion.

2. BACKGROUND

This section outlines the foundational concepts of cloud computing and resource management, providing the necessary insight for understanding the challenges and approaches discussed in this study.

2.1. Cloud Computing

Cloud Computing is one of the groundbreaking approaches, offering flexible access to computing resources. Users can allocate and release resources as needed, paying only for what they use. Its adaptability and scalability make it appealing to various applications. As can be seen in Figure 1, Cloud Computing consists of three service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). It also supports five deployment models: public, private, virtual private, community, and hybrid clouds [9]. Public clouds are open to various users, while private clouds are exclusively for one enterprise. Virtual Private Clouds are customized segments of public clouds, and community clouds serve specific industry sectors. Hybrid clouds are a combination of public and private clouds for different services [10].

The term "Cloud" signifies the wide range of resources, encompassing servers, storage, networking components, and a variety of software applications, while "Computing" optimizes the utilization of these resources, ensuring their availability to end-users based on predefined criteria in Service Level Agreements (SLAs) [6]. The abundance of accessible resources underscores the crucial role played by efficient and well-organized resource management within the cloud computing paradigm[11].

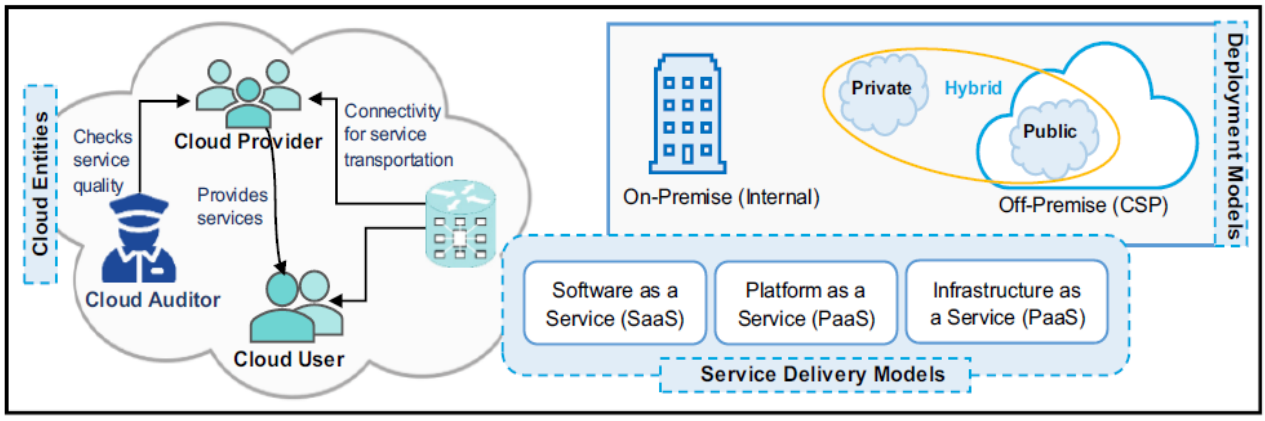

Figure 1 The Cloud Computing Model [9]

2.2. Resource Management

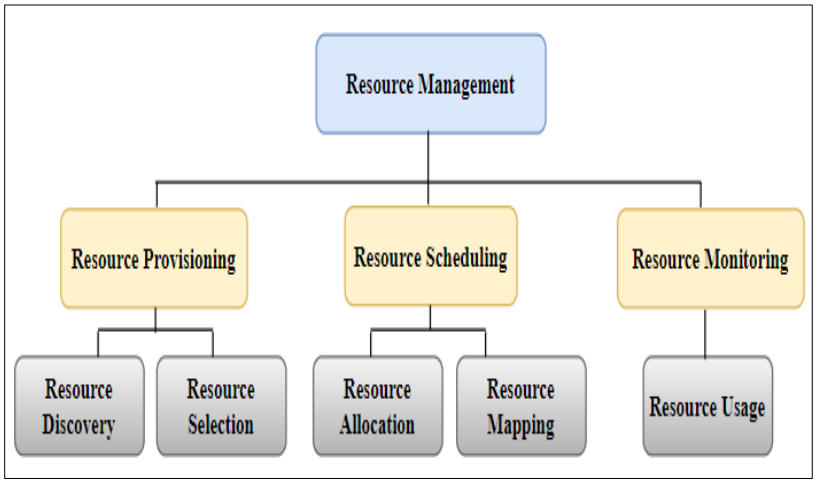

Figure 2 Taxonomy of Resource Management [12]

REVIEW ARTICLE

Effectively navigating the constantly expanding pool of resources demands skilled management strategies to ensure maximal resource utilization and system performance. Cloud service providers utilize advanced management systems to effortlessly coordinate these resources, balance the workloads, optimize performance, and enhance overall reliability [13]. As shown in Figure 2, resource management involves tasks like provisioning, scheduling, and monitoring, which all focus on meeting diverse user needs while ensuring cost-effectiveness.

2.3. Load Balancing

The fluctuating and uncertain nature of workloads in a cloud environment can result in either underutilization or overexploitation of resources, creating resource usage imbalance. This, in turn, leads to inconsistent Quality of Service (QoS), inefficient energy usage, and non-compliance with SLAs [14]. Load balancing emerges as a crucial aspect in resource management to optimize the utilization and performance of the existing resources [15]. Since each VM has unique processing speeds, storage capacities, and memory, load balancing is essential for assigning workloads to VMs so that they are not overburdened or underutilized relative to their capacity [16] [17].

2.3.1. Load Balancing Architecture

A visual representation of the workflow for load balancing in a Cloud Computing environment is depicted in Figure 3. When a user request is sent to the cloud service provider, it is analyzed and then directed to an appropriate datacenter based on resources available on that datacenter [18]. The data center controller oversees task management [16], and the load balancer takes care of distributing the workload among individual Physical Machines and their associated VMs [17]. The load balancer aims for an equitable distribution of workload across all available VMs, preventing any single VM from becoming overloaded or underloaded [18]. Meanwhile, the Virtual Machine Manager (VMM) creates and manages VMs using virtualization techniques. This balanced workload leads to improved QoS, reduced latency, and enhanced scalability, ensuring compliance with SLAs.

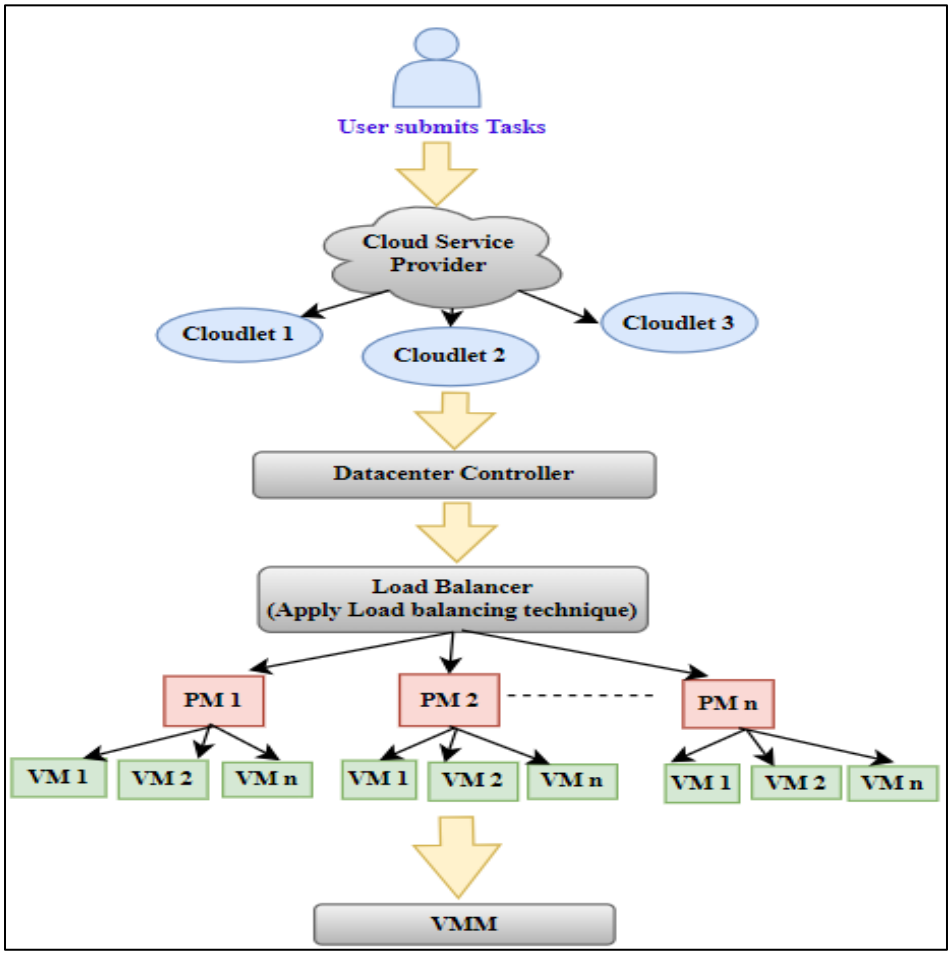

Figure 3 The Load Balancing Framework

REVIEW ARTICLE

2.3.2. Load Balancing Performance Metrics

In order to evaluate and comprehend the performance of a load balancing model, various performance metrics are used. These metrics offer critical insights into the efficacy of load balancing strategies, enabling informed decisions in resource allocation [14]. Each metric offers a distinct viewpoint on system performance. The following are the essential load balancing metrics:

- Response Time (RT)

This metric indicates the duration taken by a device to respond to a cloudlet request. A quick response is preferable to deliver a satisfactory user experience; therefore, it should be minimized.

$$RT = Start\ Time - Submission\ Time \quad (1)$$

- Makespan Time

The total time taken to allocate the available resources to all the users' requests; It should be minimized.

$$MakespanTime = Max \sum_{i=0}^{n} Completion\ Time \quad (2)$$

- Resource Utilization (RU)

It is the extent to which tasks are effectively assigned to the available resources, such that the system resources are fully utilized. It should be maximized.

$$RU = \sum_{k=0}^{n} \frac{Completion\ Time\ of\ all\ the\ jobs}{Makespan\ Time * Number\ of\ Jobs} \quad (3)$$

- Degree of Imbalance (DI)

It determines the variation in distribution of workloads among the available resources. A high degree of imbalance indicates a significant variation in the workload distribution, which can result in inefficient resource utilization, longer processing times, and potentially lower system performance, therefore, it should be minimized.

$$DI = \frac{\sigma}{\mu} \quad (4)$$

Where:

σ : standard deviation of the workload distribution.

μ : mean of the workload distribution.

- Standard Deviation

It provides insights regarding the extent to which the data points are spread around the mean, and it should be minimized.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n}} \quad (5)$$

Where:

σ: Standard deviation.

$\bar{x}$: Mean of the data points.

n : total number of data points.

- Processing Speed

It denotes the speed at which individual resources, or the overall system, processes the incoming tasks or requests. It directly impacts responsiveness and the efficiency of the system, and it must be maximized.

$$Processing\ Speed = \frac{Total\ number\ of\ tasks\ processed}{Total\ time\ taken} \quad (6)$$

- Fault Tolerance (FT)

It ensures that the algorithm maintains consistent and correct performance even when failures occur at arbitrary nodes within the system.

$$FT = 1 - \frac{Number\ Of\ Link\ failures}{Total\ Number\ of\ Links} \quad (7)$$

- Scalability

It is defined as an algorithm's ability to be able to cope with the increasing size of the network and change in requirements at any time. It should be maximized.

$$Scalabilty = \frac{New\ System\ Performance}{Initial\ System\ Performance} * 100 \quad (8)$$

- Migration Time (MT)

The time that is taken to move a given task from an overloaded server to a less loaded server. It should be minimized.

$$MT = Time\ at\ Underlaoded\ Server - Time\ at\ Overloaded\ Server \quad (9)$$

- SLA Violation

These violations may be caused when resources are overloaded and allocated VM is unavailable. It should be minimized.

$$SLA\ Violation\ Rate = \frac{Number\ of\ SLA\ Violations}{Total\ Number\ of\ Requests} * 100 \quad (10)$$

Optimally satisfying all the aforementioned parameters is crucial to satisfy the goal of achieving load-balancing, i.e. to maximize the performance of the system [19]. This research study focuses on testing parameters that include Response Time, Makespan Time, Degree of Imbalance, Resource utilization, Standard Deviation, and Processing Speed.

2.4. Taxonomy of Load Balancing Algorithms

The performance of a cloud computing environment is directly influenced by the choice and efficiency of the algorithm employed. Figure 4 depicts the taxonomy of

REVIEW ARTICLE

common load balancing algorithms utilized in cloud computing.

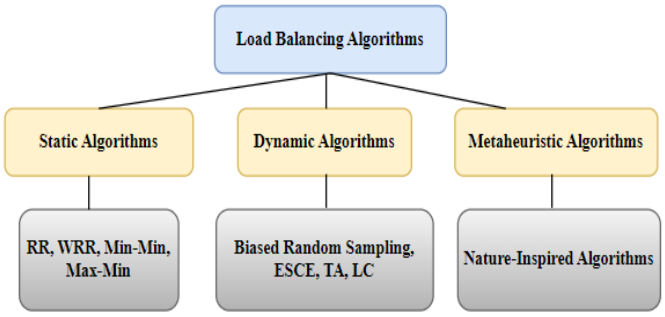

Figure 4 Taxonomy of Load Balancing Algorithms

2.4.1. Static Algorithms

These algorithms work in a fixed environment, using pre-defined information about the system's state, including factors like memory, storage, and processing power. These algorithms don't adapt to evolving system dynamics [16]. Static load balancing techniques are beneficial in scenarios where the workload remains predictable and experiences minimal fluctuations. Some of the widely utilized static algorithms for load balancing are discussed as follows:

- Round Robin (RR) Algorithm

This algorithm employs a scheduling scheme that is triggered by time to distribute tasks among machines. The time is divided in the form of slices or quanta. When consumers submit job requests to the cloud system, those requests are directed to the data center console for systematic task assignment [20].

- Weighted Round Robin (WRR) Algorithm

It is an extension of the RR approach where each process is assigned a weight proportional to its capacity. A system table is used to record and manage the weighted list of servers. The primary principle is to address tasks with heavier workloads by assigning them to robust VMs.

- Min-Min Algorithm

This algorithm uses minimal completion time for allocating resources to user requests. Thus, it selects an available machine that offers the shortest estimated completion time for pending jobs. A system table tracks the system's state and relevant node details. This allocation process proceeds iteratively until all pending tasks are handed over to a suitable VM.

- Max-Min Algorithm

This algorithm resembles the Min-Min approach in identifying machines with the least completion time for tasks. However, this algorithm focuses on reducing the completion time, especially for larger tasks. It also monitors the ready and waiting times and this process persists until all tasks are appropriately mapped to a VM.

The benefits and challenges of reviewed static load balancing methods are presented in Table 2.

Table 2 Static Load Balancing Methods

Ref. No.	Key Methods	Benefits	Challenges
[20]	Round Robin	User-friendly with improved accuracy	Increased energy consumption and longer processing times.
[1]	Weighted Round Robin	Allows for non-uniform load distribution.	Dependent on VM design and CPU usage.
[9]	Min- Min	Effectively utilizes underutilized resources.	Increased energy consumption
[21]	Max- Min	Reduces load on overloaded systems.	Increased energy consumption

2.4.2. Dynamic Algorithms

Unlike static algorithms, these algorithms are designed to adapt to fluctuating workloads and resource availability in real-time. They continuously monitor the system, making instant redistribution decisions to optimize resource use and load balancing, thus improving fault tolerance and scalability [22]. Some widely recognized dynamic load balancing algorithms are:

- Biased Random Sampling Method

Utilizes random sampling across all nodes, treating servers as nodes, and constructs a virtual graph illustrating load on each node. The load balancer then assigns the most suitable VM to fulfill user requests based on this virtual graph.

- Equally Spread Current Execution (ESCE)

Ensures equitable workload distribution among servers by prioritizing tasks, evaluating their size and capacity, and identifying the server that is most efficient to handle the load. Tasks are assigned intelligently based on both VM capacity and estimated load.

- Throttled Algorithm (TA)

It maintains index metrics for VMs, considering factors such as processing speed, capacity, and storage, along with monitoring how busy the VMs are. A user computer sends a

REVIEW ARTICLE

request to the cloud data center, and the most suitable VM is selected based on indexed metrics and VMs' current state.

- Least Connection (LC) Method

Redistributes workload by choosing the server with the fewest active transactions. Dynamic scheduling directs user requests to the chosen cloud server at the time of the user's request [14].

The benefits and challenges of reviewed static load balancing methods are presented in Table 3.

Table 3 Dynamic Load Balancing Methods

Ref. No.	Key Methods	Benefits	Challenges
[20]	Biased Random Sampling method	Enhanced precision and resource utilization.	Increased energy consumption
[23]	ESCE	Fair and efficient distribution of workloads.	Challenges in accurately estimating future loads and server capacities.
[24]	TA load balancing method	Improved utilization of resources.	Performs optimally under similar workload conditions.
[25]	LC method	Enhanced accuracy and performance.	Requires additional processing time.

2.4.3. Metaheuristic Algorithms

Metaheuristic algorithms play an instrumental role in addressing dynamic and complex challenges in cloud computing, particularly in areas like resource allocation, scheduling, and load balancing. Conventional optimization approaches often struggles with the computational complexity of these tasks, necessitating the adoption of metaheuristics [26].

The main features of Metaheuristic Algorithms that contribute to their popularity are as follows:

- Inspired by Nature

They draw inspiration from various scientific, biological, and evolutionary principles.

- Dynamic Load Balancing

These algorithms adapt to fluctuating resource demands in dynamic cloud environments, preventing resource overloading and ensuring efficient utilization.

- Optimization of Resource Allocation

By considering factors like resource capacity, task requirements, and network conditions, these algorithms allocate resources to reduce response time, maximize utilization, or achieve other objectives.

- Multi-Objective Optimization

They simultaneously handle multiple objectives that are conflicting in nature. Thus, balancing considerations like reducing response time and consumption of energy, while also enhancing resource utilization, all at once.

- Adaptability

They adapt to changes in the working environment and continuously search for improved solutions.

- Scalability

Designed to handle large-scale optimization problems, they effectively manage massive cloud environments with thousands or millions of resources.

- Stochastic Nature

The search process incorporates randomness, which allows it to overcome local optima and examine diverse areas of the solution space.

- Exploration and Exploitation

These are the fundamental components of Metaheuristic Algorithms.

- Exploratory Diversification

Utilizes random sampling to uncover optimal solutions, preventing local optima traps and fostering diversity in iterations.

- Exploitative Intensification

Selects the best solution from the nearby neighborhood, aiding convergence [27].

2.5. Metaheuristic Algorithms for Load Balancing

In the dynamics of cloud computing, metaheuristic algorithms provide versatile strategies to efficiently utilize resources, enhance system responsiveness, and ensure reliability through dynamic adaptation to changing conditions. Inspired by these metaheuristic principles, load balancing algorithms consider various factors such as the health of the server, task execution time, and resource availability. These algorithms, inspired by nature, are categorized as Nature-Inspired Algorithms (NIAs), which include Bio-Inspired (BI) Algorithms like Swarm Intelligence (SI) and Evolutionary algorithms. The taxonomy of Metaheuristic Load Balancing Algorithms is depicted in Figure 5. It also shows further categorization of load

REVIEW ARTICLE

balancing algorithms that are inspired by physics and chemical processes, including the Water Cycle Algorithm and Black Hole Optimization. Additionally, some algorithms draw inspiration from domains like music, emotions, and social interactions [26].

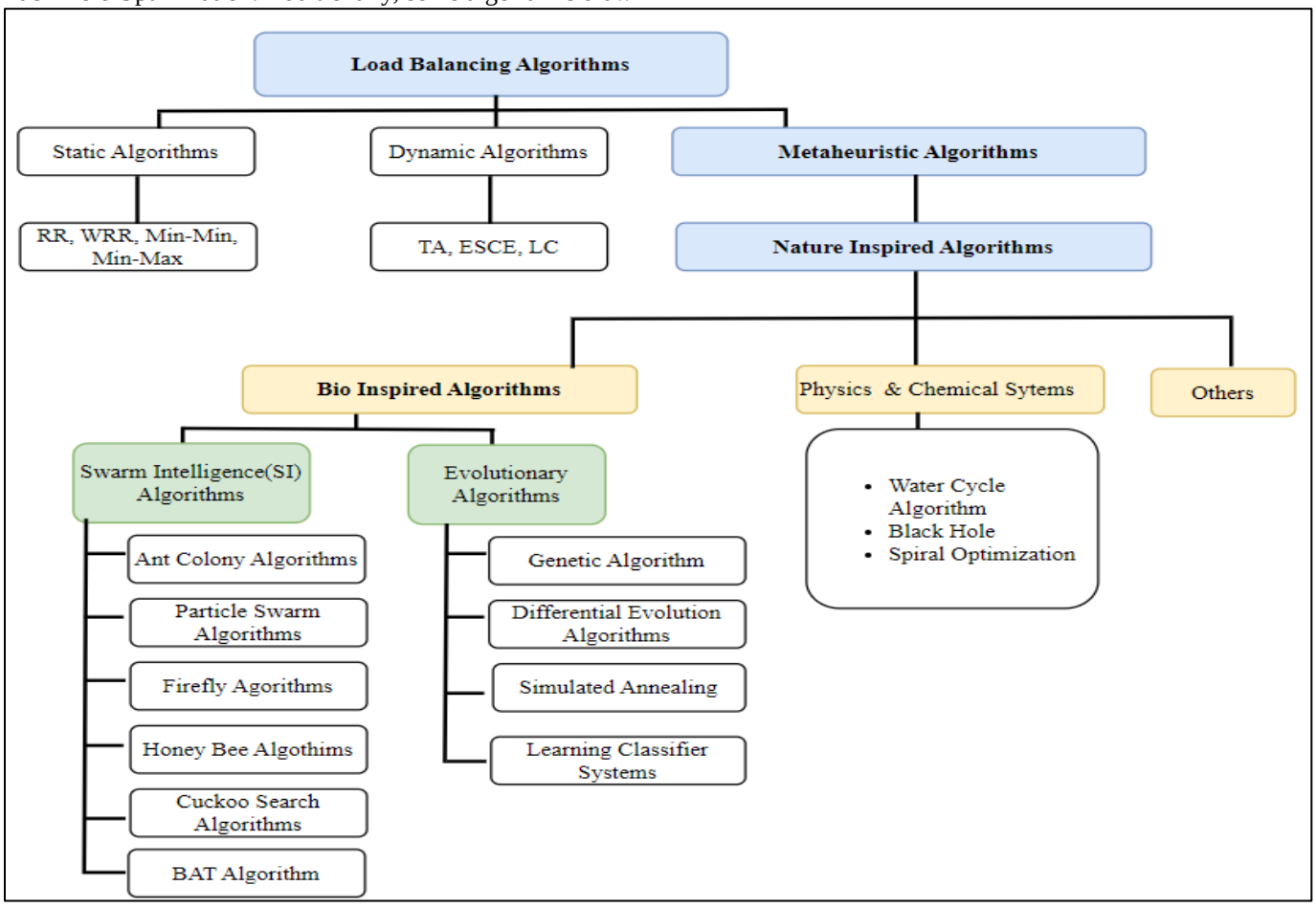

Figure 5 Taxonomy of Metaheuristic Load Balancing Algorithms

3. LITERATURE REVIEW

Bio-inspired metaheuristic algorithms have garnered significant interest within the research community because of their ability to be inspired by biological processes to address complex optimization problems. Their adaptability in dynamic and uncertain settings makes them highly relevant for diverse real-world applications. The research focus has been narrowed down to Bio-Inspired Swarm Intelligence algorithms.

These algorithms present an alluring prospect for tackling intricate optimization and searching for solutions through nature's adaptive strategies. By thoroughly analyzing Swarm Intelligence algorithms, this study seeks to gain a profound understanding of their application, strengths, and weaknesses, thereby facilitating a more informed and effective approach to research efforts. The review is done in groups of Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly Optimization (FA), and the BAT Algorithm.

3.1. Ant Colony Optimization (ACO)

ACO is another metaheuristic approach inspired by Marco Dorigo's Ant System (AS) [28]. This algorithm mimics the foraging behavior of ants, which locate the shortest route between their nest and a food source. As a pioneering swarm intelligence-based algorithm, it is useful for tackling challenging Combinatorial Optimization (CO) problems. The working of an ACO to look for the shortest and optimal path between food sources and their colonies is illustrated in Figure 6. It shows how a swarm of artificial ants navigates through the solution space by depositing pheromone trails and selecting paths with the highest pheromone concentration. The ants follow the selected path, and the pheromone gradually evaporates. This iterative process persists until the best solution is found, i.e., reaching the food source following

REVIEW ARTICLE

the shortest route. ACO has been extensively employed in load balancing for cloud environments, owing to its efficacy in analyzing intricate solution spaces. Due to its ability to understand intricate solution spaces, ACO has found extensive application in optimizing load distribution within cloud environments [17]. In the ACO-based load balancing, tasks are represented by ants, while pheromone trails serve as the indicators of the resource utilization associated with each node. The algorithm is designed to adaptively modify the pheromone trails for optimal distribution of tasks among the available resources [29]. A comprehensive study of ACO algorithm literature has revealed its versatile applications across various domains and its significant contributions to advancing load balancing and resource management within cloud computing. Table 4 summarizes the key findings, strengths, and weaknesses of the reviewed algorithms. It was observed that ACO-based load balancing literature emphasizes improving metrics like resource utilization, throughput, response time, and makespan. The drawbacks of ACO include scalability issues leading to suboptimal load distribution. Also, limited fault tolerance and increased computational demands can raise energy consumption.

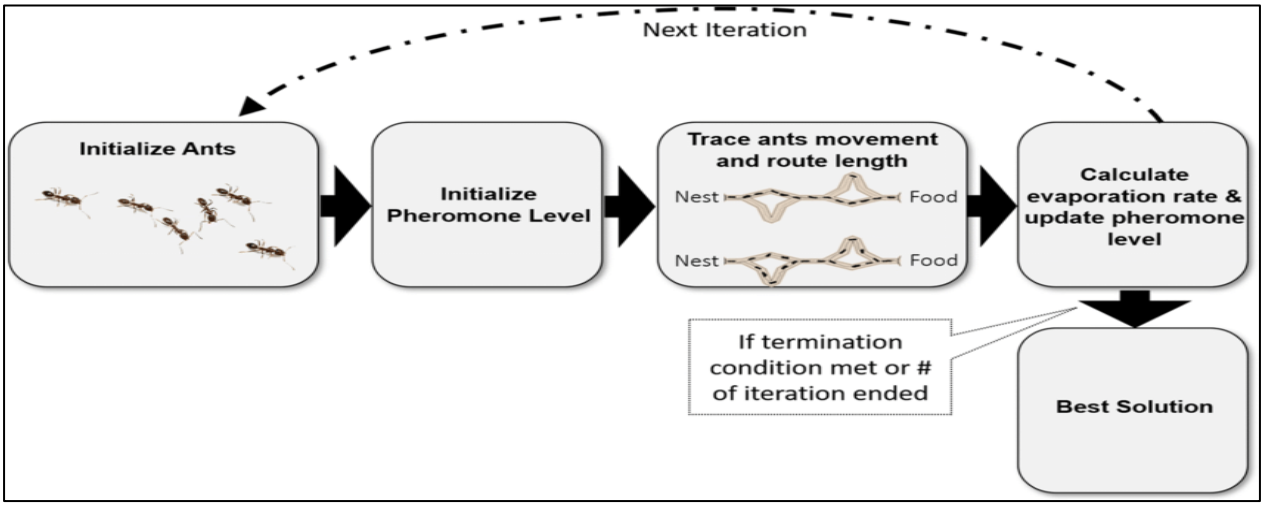

Figure 6 Flowchart Depicting the Working of ACO

Table 4 Review of ACO-Based Load Balancing Algorithms

Ref. No.	Algorithms	Key Findings	Strengths	Weaknesses
[30]	SALB	• Rapidly identify overloaded nodes by utilizing ants to create a pheromone table.	• High resource utilization • High throughput • Low response time • Low overhead • Less energy consumption	• Low fault tolerance • High makespan • Low performance
[31]	ACO to detect overloaded or underloaded nodes	• Identifies under-loaded and overloaded servers using foraging and trailing pheromones.	• High availability • High resource utilization • High throughput • Low response time	• Restricted to homogeneous environment • High makespan • Low scalability
[32]	ACO-VMM	• Considers system history to prevent unnecessary migrations. • Utilizes two distinct ant traversal strategies	• High reliability • High scalability • Low migrations • Low makespan • Low response time	• Low fault tolerance • High energy consumption

REVIEW ARTICLE

[33]	VMDPS-ACO	• Uses ACO for dynamic prediction scheduling to forecast and prevent memory overloads.	• High resource utilization • Low response time • Less Migrations	• High execution time • High response time • Low reliability • Low QoS
[34]	Ant Lion Optimizer (ALO)	• Addresses cloud VM scheduling through task submission, scheduling, and VM processing phases. • Considers both makespan and total task execution time in the cost function.	• Low makespan • Low execution time	• Low scalability • Low QoS • High overhead • Evaluation only in the small Environment
[35]	Enhanced ACO	• Employs random search to assign jobs to VMs. • Aims to identify tasks with minimal overhead and suitable resources.	• Low makespan • Low cost • Low number of migrations • High resource utilization • High throughput	• High energy consumption • Low scalability

3.2. Particle Swarm Optimization (PSO)

In 1995, the PSO metaheuristic technique was introduced, which was inspired by the coordinated social behavior exhibited by flocks of birds or schools of fish. It is a stochastic approach that is population-driven and designed to address both continuous and discrete optimization problems. It orchestrates a collective movement of particles within a search space to explore and identify optimal solutions [36]. While its initial application was in neural network training, PSO has since found utility across a diverse array of domains, including telecommunications, data mining, design, control systems, combinatorial optimization, power systems, and signal processing [10]. The flowchart in Figure 7 illustrates the iterative process through which PSO evaluates candidate solutions using a fitness function. Each particle denotes a potential candidate solution, with its position and velocity rate continually adjusted based on both individual and group behavior. This collaborative process enables PSO to effectively navigate complex and dynamic search spaces. In load balancing applications, tasks are represented by particles with positions and velocities denoting resource usage levels for each node. Workload distribution across servers is achieved by adjusting particle positions and velocities, considering the best personal positions and global best positions. This iterative process persists until satisfactory solutions are found. PSO was first utilized for load balancing in a study [37], where a Task-based System Load Balancing method using Particle Swarm Optimization (TBSLB-PSO) was introduced. This approach relocates only excess tasks from an overloaded VM, rather than migrating the whole VM.

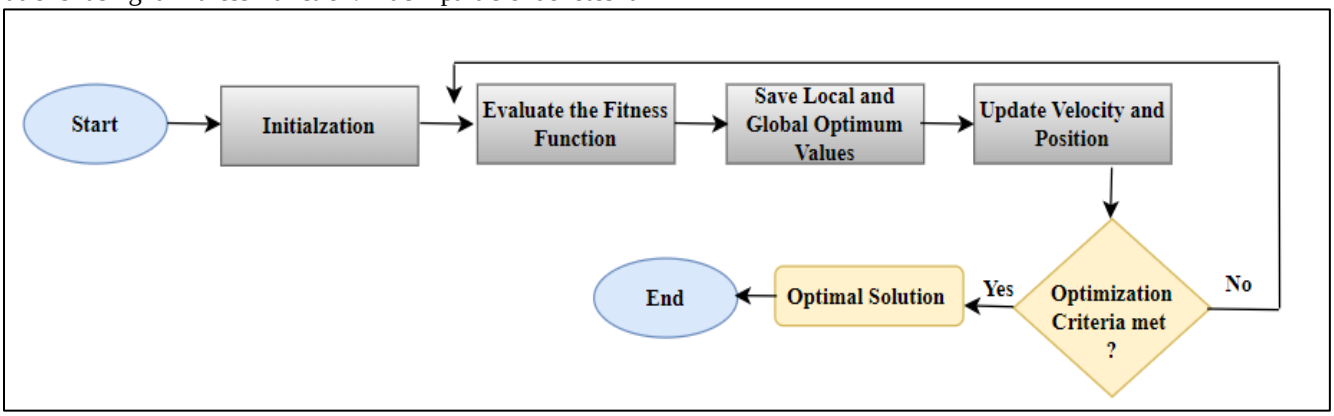

Figure 7 Flowchart of Particle Swarm Optimization [38]

REVIEW ARTICLE

Table 5 Review of PSO-Based Load Balancing Algorithms

Ref. No.	Algorithm	Key Findings	Strength	Weakness
[39]	Local Minima Jump PSO	• Tracks convergence through changes in gbest and particle velocities. • Highly effective, particularly when dealing with a larger number of tasks.	• Low makespan • High scalability	• Low fault tolerance • High execution time
[40]	VM PSO Load Balancer	• PSO-centered algorithm for server load distribution among VMs to enhance resource utilization. • Showcases reduced under-/over-utilization.	• Low response time • High resource utilization	• High makespan • Low reliability • Low QoS
[41]	Endocrine-PSO	• The endocrine algorithm draws inspiration from the human hormone regulation system. • Self-organization among overloaded VMs through a feedback approach.	• Low makespan • Low migration time • High QoS	• High response time • Low reliability
[42]	Improved PSO	• Formulates an allocation model that considers the characteristics of complex networks.	• High resource utilization • Low Imbalance Degree • Low Completion Time • Low Makespan	• Low QOS • High Energy Consumption
[43]	Improved version of PSO using a red-black tree	• Utilizes a red-black tree and discrete PSO algorithm. • The red-black tree divides tasks into queues and assigns them randomly, contributing to improved balance.	• Low Imbalance Degree • Low Makespan • Low Execution Time • High Resource utilization	• Low fault tolerance • Low scalability • Low Reliability

A comprehensive study of the PSO algorithm in existing literature has unveiled its wide-ranging utilization in diverse fields and its notable role in enhancing load balancing for efficient resource management within cloud computing.

A condensed overview of the main discoveries, advantages, and limitations of the examined algorithms is presented in Table 5. It is observed that PSO excels in achieving low makespan and efficiently minimizing the total task completion time. Additionally, the algorithm demonstrates high resource utilization and optimal resource allocation.

However, one drawback of PSO is its susceptibility to premature convergence, causing it to settle in local optima and deliver suboptimal results. The algorithm also struggles with adapting to dynamic changes, impacting scalability, and has lower fault tolerance.

REVIEW ARTICLE

3.3. Firefly Algorithm (FA)

FA is another optimization technique derived from nature, simulating the typical behavioral traits of fireflies [44]. As a swarm intelligence algorithm, it excels in solving optimization problems. The algorithm's workflow is depicted in Figure 8. The algorithm begins by initializing a population solution, evaluating fitness through an objective function, and then iteratively adjusting firefly positions based on attractiveness and distance. Fireflies exhibiting dimmer light are attracted to and move toward brighter ones. In case no brighter fireflies are discovered, then the firefly will move randomly. Fitness is updated, and convergence is checked against predefined criteria. Firefly with the highest fitness indicates the optimized solution. The adaptability and efficient exploration of solution spaces make FA an ideal choice for load balancing in cloud computing. Here, VMs or tasks act as fireflies, and brightness reflects quality. VM/task migration mimics fireflies seeking better nodes for balance. By dynamically updating firefly positions, the algorithm adapts to changing cloud conditions, promoting balanced resource use, preventing overload, and enhancing system efficiency.

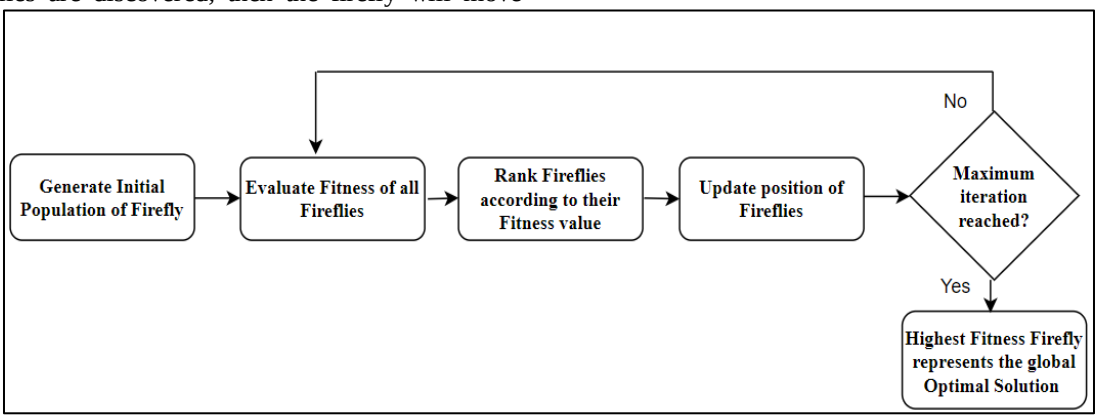

Figure 8 Flowchart of Firefly Algorithm [45]

Table 6 Review of FA-Based Load Balancing Algorithms

Ref. No.	Algorithm	Key Findings	Strengths	Weaknesses
[46]	Proposed FA algorithm	• Three-phase approach: population generation, scheduling indices calculation, and scheduling list optimization.	• Low execution time • Low process time • High resource utilization	• Heterogeneous workloads not considered • Low QoS • Low scalability • Low reliability
[47]	Novel FA algorithm	• A novel approach with fireflies as candidates for optimizing makespan and flow time. • Jobs are allocated to virtual machine resources based on job size, length, and processing speed.	• Low makespan • Low flow time	• Low scalability • Low resource utilization • Low QoS
[48]	Firefly Load Balancer	• Threshold value set for virtual nodes to prevent exceeding processing capacity	• Low response time • High resource utilization • Low energy consumption • High throughput	• Low QoS • Independent tasks • Low reliability
[49]	Proposed FA algorithm	• Each virtual machine comprises three servers, each with three nodes, and their characteristics are determined through the scheduling process. • Identifies less busy nodes and adds them to the cloud to assist with tasks.	• High load-balancing • High flexibility • Low execution time • Low makespan	• High energy consumption • Low QoS

REVIEW ARTICLE

A summary of the important findings, strengths, and related weaknesses of the FA-based algorithms is presented in Table 6. The review helps understand that the FA algorithm proves highly effective in cloud load balancing, demonstrating strengths such as low execution time, process time, and makespan. This leads to optimal resource utilization and energy efficiency, along with impressive throughput and load-balancing capabilities. However, it must be noted that FA may not address heterogeneous workloads, potentially impacting Quality of Service (QoS). Scalability, reliability, and resource utilization may present additional challenges.

3.4. BAT Algorithm

Drawing on the echolocation abilities of bats, a novel optimization approach, called the BAT algorithm, was introduced in 2010 [50]. During their pursuit of prey, bats utilize this ability to estimate the distance to the prey, with continuously adjusting their frequency, loudness, and pulse rate. This adaptive behavior inspires the BAT algorithm, which shares similarities with the PSO algorithm in terms of the change in velocities and positions. Essentially, this algorithm can be viewed as a fusion of PSO and intensive local exploration, with constraints based on loudness and pulse emission rate. The working flowchart of the BAT algorithm is depicted in Figure 9. The workflow begins with the initialization of a population of virtual bats, each assigned random values of positions, velocities, frequencies, and loudness. Bats iteratively adjust their positions and velocities in response to their current condition and the optimal solutions discovered so far. Additionally, bats may undergo local search operations, where they explore new positions around the best and worst solutions using random perturbations. As the algorithm progresses, the bats' frequencies and loudness gradually decrease. Termination occurs when the specified convergence condition is fulfilled. The interplay between exploration, which is searching for new possibilities, and exploitation, which is enhancing promising solutions, guides the bats toward high-quality solutions. The algorithm integrates exploration, adaptation, parallelism, and global optimization, making it particularly suitable for load balancing within cloud computing. Inspired by bats' hunting behavior, it aids in identifying machines with lower loads. The algorithm prioritizes searching for the optimal VM among all available ones when assigning tasks, thus ensuring efficient resource utilization [47]. A study [48] showcased the BAT algorithm's effectiveness in VMs, affirming its ability to enhance load balancing. It addresses the challenges posed by dynamic workloads, diverse resource configurations, scalability demands, and performance goals, thereby ensuring optimal resource utilization, improved system responsiveness, and an enhanced user experience in cloud-based applications [51][52].

A detailed literature survey of the BAT algorithm highlights its wide-ranging applications across various domains and its significant contributions to advancing resource management and load balancing. The key findings, along with strengths and limitations of the algorithm, are summarized in Table 7. The review shows that the BAT algorithm proves highly effective in cloud load balancing, demonstrating strengths such as low execution time, process time, and makespan. This leads to optimal resource utilization and energy efficiency, along with impressive throughput and load-balancing capabilities. However, it's important to note that BAT may not address heterogeneous workloads, potentially impacting Quality of Service (QoS). Scalability, reliability, and resource utilization may present additional challenges.

Table 7 Review of BAT algorithm-Based Load Balancing Algorithms

Ref. No.	Algorithm	Key Findings	Strengths	Weaknesses
[53]	Improved BAT Optimization (IBO)	• Multi-server environment. • Ensures all servers remain consistently utilized, avoiding under or overloading.	• Response Time is minimized. • Facilitates load balancing without delays.	• Job migrations impact response time.
[54]	A hybrid of PSO and BAT	• Utilizes PSO for task placement and BAT for VM migration. • Considers data center bandwidth, CPU, and memory.	• Reduces energy consumption and heat dissipation.	• Security concerns are not considered during task migrations.
[55]	Micro BAT	• Pre-classifies tasks based on resource requirements. • Incorporates elasticity in VM allocation and manages task migration by adding a local queue to prevent server overload.	• Reduced response time with minimal delays. • Elasticity and queue management.	• Migrations increase execution time. • Each VM class should have similar characteristics.

REVIEW ARTICLE

[56]	Improvised BAT Algorithm	• Manages cloud-based IoT load, utilizing load distribution methods for population generation. • It prioritizes tasks with the shortest execution time.	• Low Execution Time	• Inability to effectively handle large-scale scenarios.
[57]	Load Balancing Bat Algorithm (LBBA)	• Utilizes a Naive-Bayes classifier for VM categorization and task migration.	• Low Response, Completion and Migration Time • Resource Utilization is better.	• Task priority is the only QoS parameter considered.
[58]	Modified Bat Algorithm (MBA)	• The algorithm works in two variants: MBA with Overloaded Optimal Virtual Machine (MBAOOVM) and Modified Bat Algorithm with Balanced Virtual Machine (MBABVM). • Targets VMs with higher energy levels for task allocation.	• Reduces Execution Time. • Energy Efficient • Cost-effective.	• Scalability issues, as increasing bat numbers does not enhance results

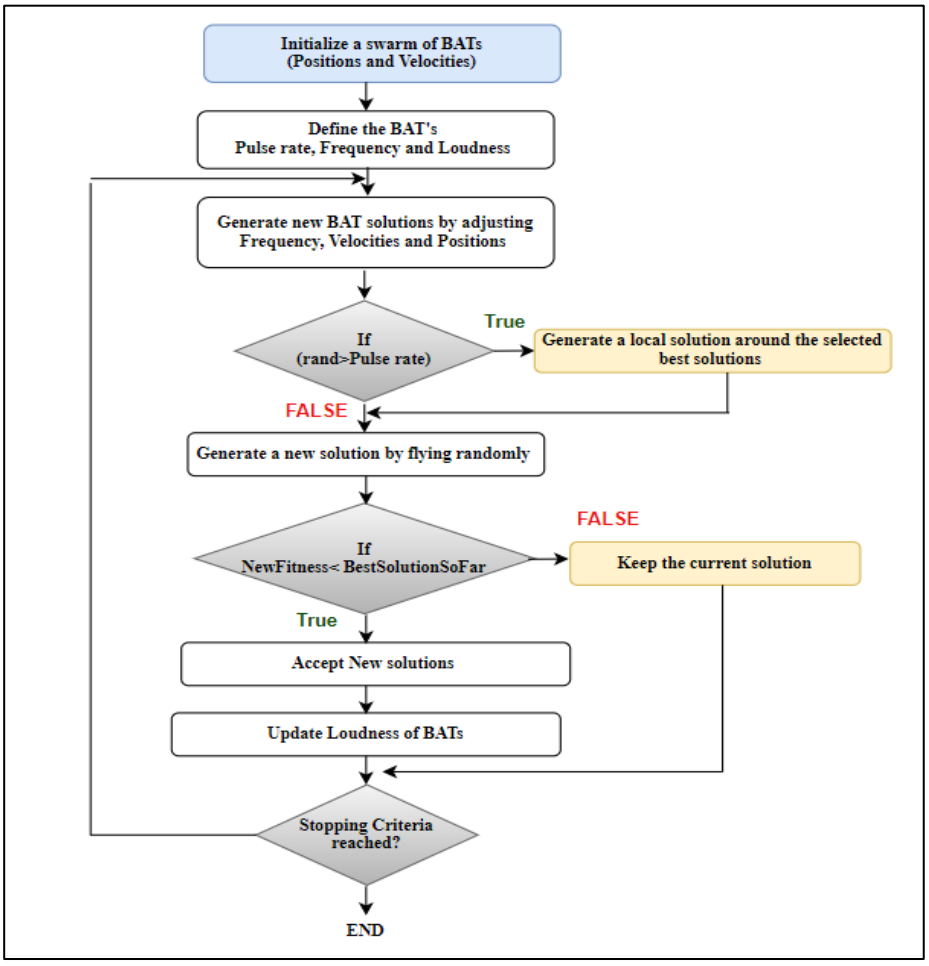

Figure 9 Flowchart of BAT Algorithm [51]

REVIEW ARTICLE

4. PERFORMANCE EVALUATION

The current section analyzes the obtained results and assesses the performance of the reviewed swarm intelligence metaheuristic load balancing methods, which were implemented in the CloudSim simulator using Java. A simulation library based on CloudSim facilitated their implementation, which provides numerous methods for replicating diverse simulation scenarios [59]. The assessed algorithms encompass ACO, PSO, FA, and BAT algorithms. These selections were made based on their recognized prominence and efficacy in addressing load-balancing complexities within the realm of bio-inspired swarm intelligence metaheuristics.

4.1. Experimental Setup

The experimental environment was established using CloudSim, which encompassed the configuration of datacenters as represented in Table 8, host machine configurations as represented in Table 9, and virtual machines (VMs) configuration as represented in Table 10. The number of tasks or cloudlets ranged between 100 to 500.

Each cloudlet had varying lengths representing the varying workloads in a dynamic cloud environment. The performance parameters considered for this investigation included response time, makespan, degree of imbalance, resource utilization, standard deviation, and processing speed.

Table 8 Datacenter Configuration

Environment Setup	
Servers	
Specs \ Name (id)	Server 0
Architecture	x64
Operating System	Linux
VM Monitor (VMM)	Xen
Time Zone	10.0
Second Cost	3.0
Memory Cost	0.05
Bandwidth Cost	0.0
Storage Cost	0.001
Scheduling Interval	0.0
Hosts (Per Clone)	0
Clones	2
Random Style (if any)	Fixed_Pace

Table 9 Host Configuration

Hosts	
Specs \ Host(id)	Host 0
MIPS	177730.0
Processing Elements	6
Memory (RAM in MB)	16000
Bandwidth (in MB/s)	15000
Storage (in MB)	4000000
Clones	2
Random Style (If Any)	Fixed_Pace

Table 10 Virtual Machine Configuration

Virtual Machines	
Specs \ VM (id)	VM Type 0
VM Monitor (VMM)	Xen
MIPS	15000.0
Processing Elements	2
Memory (RAM in MB)	1024
Bandwidth (in MB/s)	1000
Image Size (in MB)	7000
Clones	3
Random Style (If Any)	Fixed_Pace

4.2. Results and Discussions

This section reports the results obtained from executing the algorithms in the Cloud Simulator environment. Figure 10 depicts the CloudSim logs pertaining to the Swarm Intelligence Metaheuristic load balancing techniques. The algorithms were tested across ranges of 100 to 500 cloudlets. Figure 11(a) and Figure 11(b) depict the results corresponding to 100 and 500 cloudlets, respectively, for each of the observed algorithms.

REVIEW ARTICLE

CloudSim Simulation Log

```
Initialising...
Starting CloudSim version 3.0
koala(0) is starting...
Server[0] is starting...
Server[1] is starting...
Entities started.
0.0: koala(0): Cloud Resource List received with 2 resource(s)
0.0: koala(0): Trying to Create VM #0 in Server[0]
0.0: koala(0): Trying to Create VM #1 in Server[0]
0.0: koala(0): Trying to Create VM #2 in Server[0]
0.0: koala(0): VM #0 has been created in Datacenter #3, Host #0
0.1: koala(0): VM #1 has been created in Datacenter #3, Host #1
0.1: koala(0): VM #2 has been created in Datacenter #3, Host #0
49.0276: koala(0): Cloudlet 33 received
50.8538: koala(0): Cloudlet 72 received
51.109: koala(0): Cloudlet 88 received
51.219: koala(0): Cloudlet 63 received
52.779: koala(0): Cloudlet 62 received
```

Figure 10 CloudSim Logs for Swarm Intelligence Metaheuristic Load Balancing Methods

REsearchers Cloud

Particle Swarm Optimization By github.com/cypherskar 100.000%

Sending Tasks 100.000% | Recieving Tasks 100.000%

Experiment					
Simulation for					100-Tasks
Algorithm	(id)	ACO (0)	Bat (1)	FA (2)	PSO (3)
Response Time	(ms)	26.8723	101.5764	101.5767	30.0607
Makespan	(ms)	39.1328	101.6764	101.6767	47.5024
Vms Makespan	(ms)	102.0934	101.6764	101.6767	102.1125
Resource Utilization		0.2642	1	1	0.2954
Imbalance Degree		0.3573	0.2363	0.2363	0.6773
Standard Deviation		1.1742	3.6118	3.6118	0.9368
ProcessingSpeed (ms)		88	296	2,576	4,062
Vm Scheduler		TimeShared	TimeShared	TimeShared	TimeShared
Task Scheduler		TimeShared	TimeShared	TimeShared	TimeShared

Distribution Matrix			TasksSplit.Even
Broker	(id)	Task Type (0)	Total
koala	0	100	100
Total		100	100

Figure 11(a) Results for Swarm Intelligence Metaheuristic Load Balancing Methods for 100 Tasks

Simulation for					500-Tasks
Algorithm	(id)	ACO (0)	Bat (1)	FA (2)	PSO (3)
Response Time	(ms)	158.1522	492.2722	492.2673	190.7731
Makespan	(ms)	173.6722	492.3722	492.3673	190.8731
Vms Makespan	(ms)	493.1376	492.3722	492.3673	503.232
Resource Utilization		0.0642	0.2	0.2	0.0759
Imbalance Degree		0.0938	0.0488	0.0488	0.3594
Standard Deviation		1.1231	7.948	7.948	2.1307
ProcessingSpeed (ms)		48	368	9,208	17,230
Vm Scheduler		TimeShared	TimeShared	TimeShared	TimeShared
Task Scheduler		TimeShared	TimeShared	TimeShared	TimeShared

Distribution Matrix			TasksSplit.Even
Broker	(id)	Task Type (0)	Total
koala	0	500	500
Total		500	500

Figure 11(b) Results for Swarm Intelligence Metaheuristic Load Balancing Methods for 500 Tasks

REVIEW ARTICLE

4.2.1. Response Time

Figure 12 compares the algorithms on response time, which, as shown in equation (1) of Section 2.3.2, is calculated as the difference between a cloudlet's start and submission times. The results indicate that ACO consistently achieves the lowest response time across all cloudlet counts, with PSO close behind. By contrast, FA and BAT record noticeably higher response times and perform at a similar level to each other.

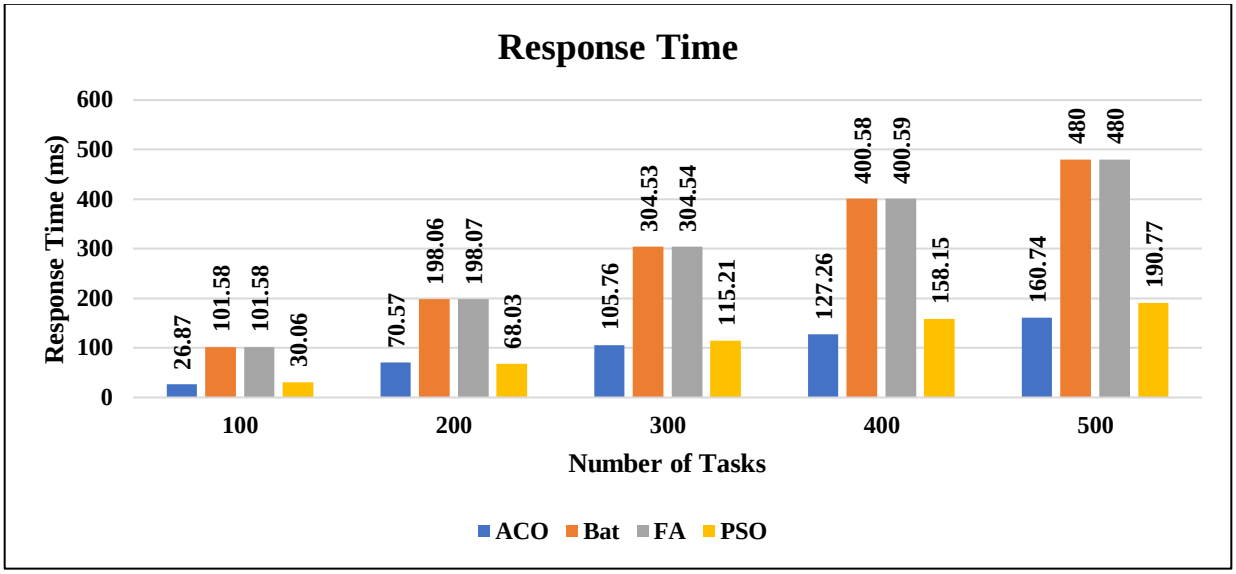

Figure 12 Response Time (ms)

4.2.2. Makespan Time

Figure 13 illustrates a comparison of makespan times across different task volumes, evaluated using equation (2). Similar to response time analysis, ACO achieved the shortest Makespan Time, followed by PSO. The FA and BAT algorithms exhibited longer makespan times, showing comparable performance in this aspect also.

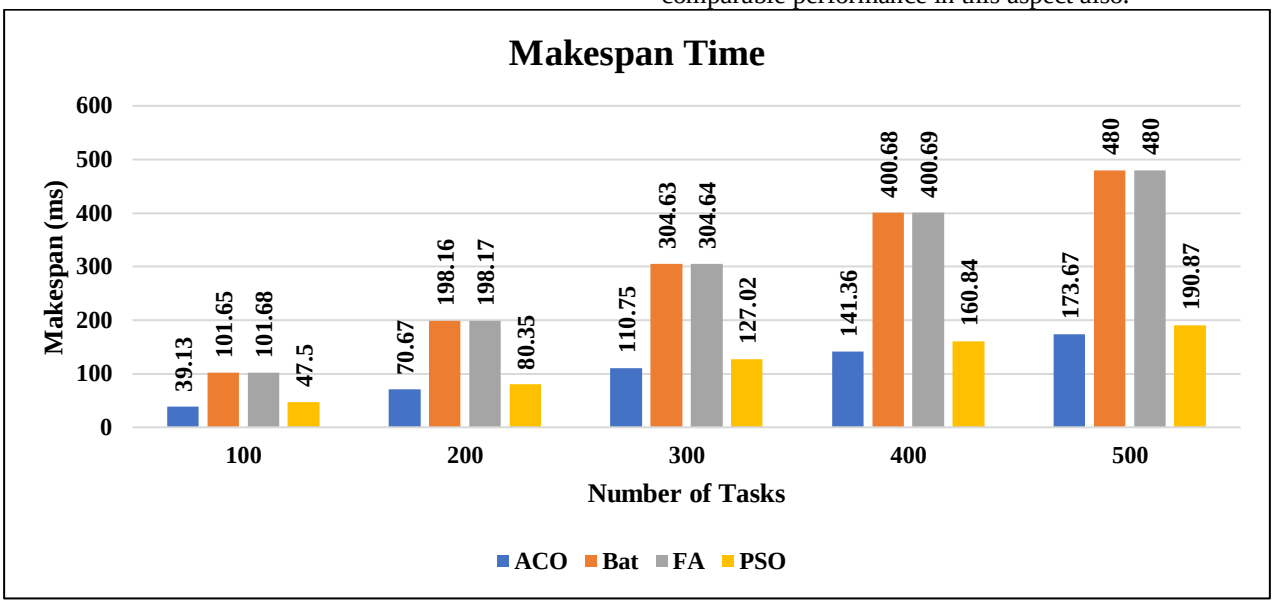

Figure 13 Makespan Time (ms)

4.2.3. Resource Utilization

Figure 14 illustrates a comparison of load balancing algorithms in terms of resource utilization, evaluated using equation (3). The results indicate that the FA and BAT algorithms achieve higher resource utilization than ACO and PSO algorithms.

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2025/36 Volume 12, Issue 4, July – August (2025)

REVIEW ARTICLE

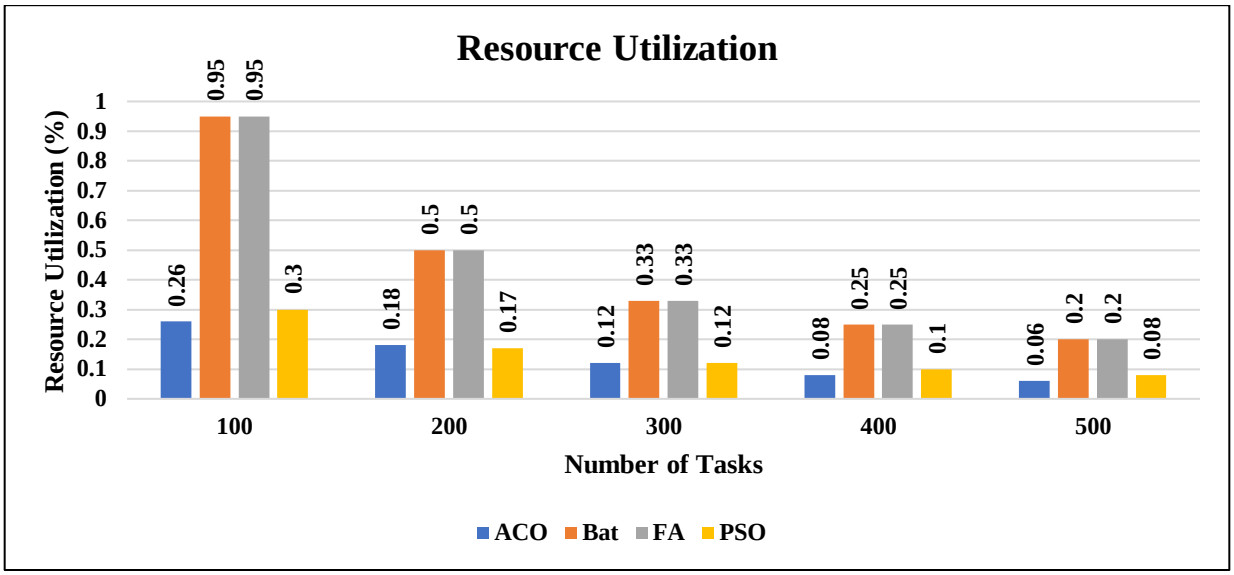

Figure 14 Resource Utilization (%)

4.2.4. Degree of Imbalance

Figure 15 shows a comparison based on the Degree of Imbalance. Equation (4) is utilized for the evaluation of algorithms. Notably, PSO exhibited the highest degree of imbalance, indicating a significant disparity in workload distribution. ACO gave improved performance compared to PSO, presenting reduced imbalance. Conversely, FA and BAT demonstrated the lowest degree of imbalance, indicating a more uniform workload distribution across resources compared to other algorithms.

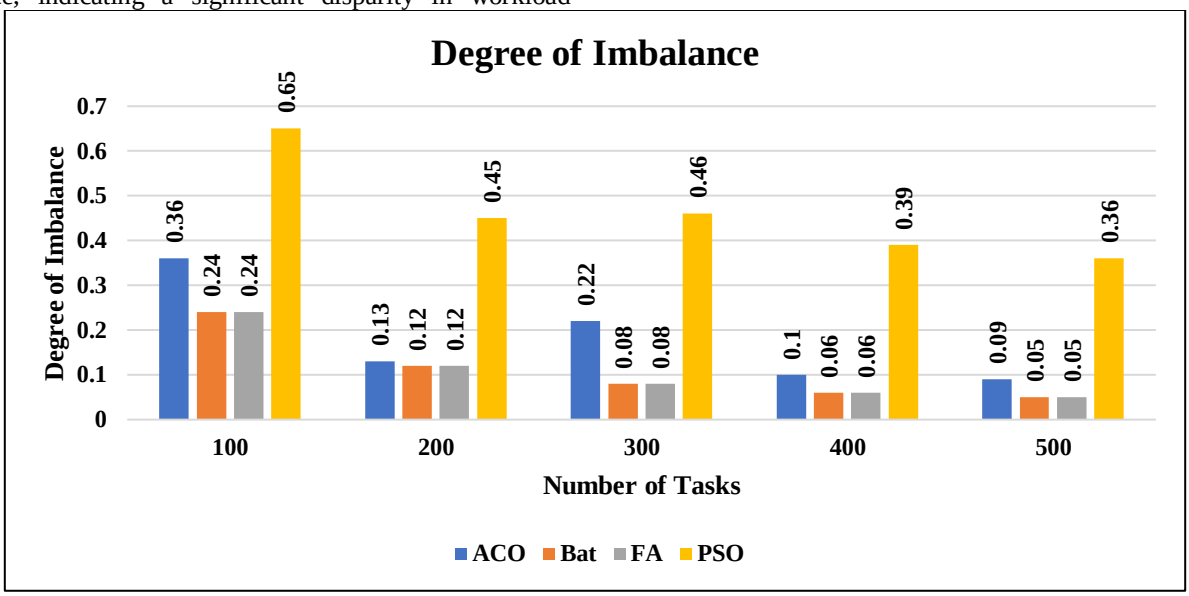

Figure 15 Degree of Imbalance

4.2.5. Standard Deviation

Comparison based on Standard Deviation is depicted in Figure 16, where equation (5) is utilized for evaluation, and it shows that ACO consistently gave the lowest standard deviation, indicating tightly clustered data points around the mean. PSO followed with slightly higher values. However, FA and BAT algorithms exhibited higher standard deviations, suggesting greater variability in their data distribution. It's essential to distinguish between DI and standard deviation. While both metrics offer insights into data distribution, DI specifically addresses the uneven distribution of workload or resources whereas standard deviation offers a broader measure of data variability around the mean.

REVIEW ARTICLE

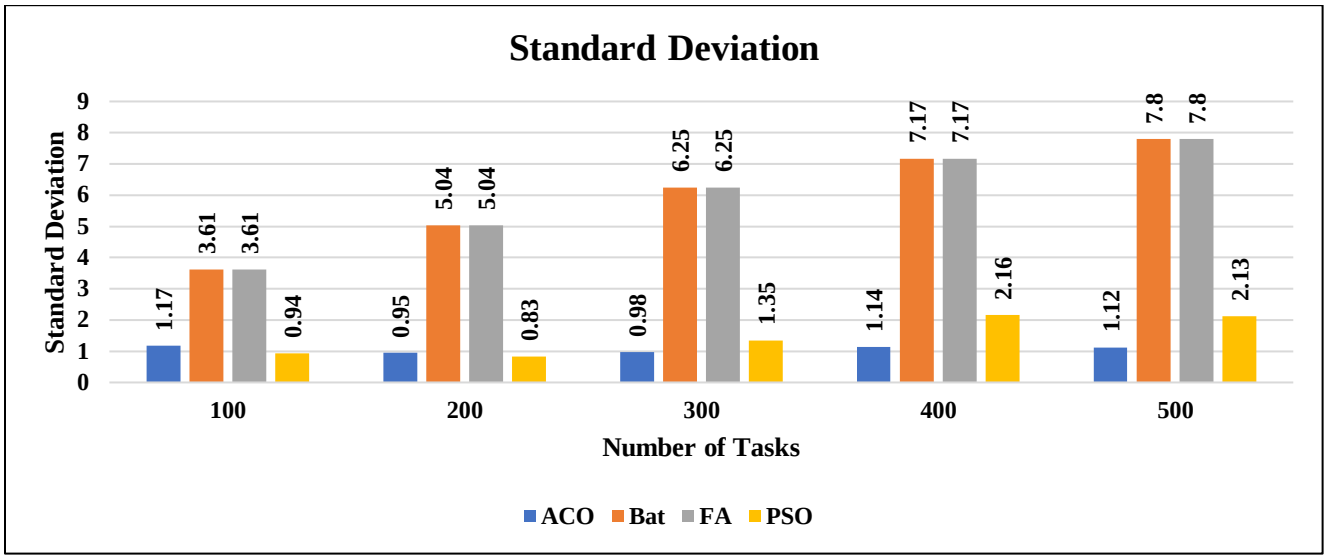

Figure 16 Standard Deviation

4.2.6. Processing Speed

Figure 17 compares load balancing algorithms based on their processing speeds. Equation (6) is utilized for evaluation, and the findings suggest that the PSO algorithm emerged as the top performer in terms of processing speed, with FA following closely behind. Conversely, both the BAT and ACO algorithms demonstrated comparatively slower processing speeds.

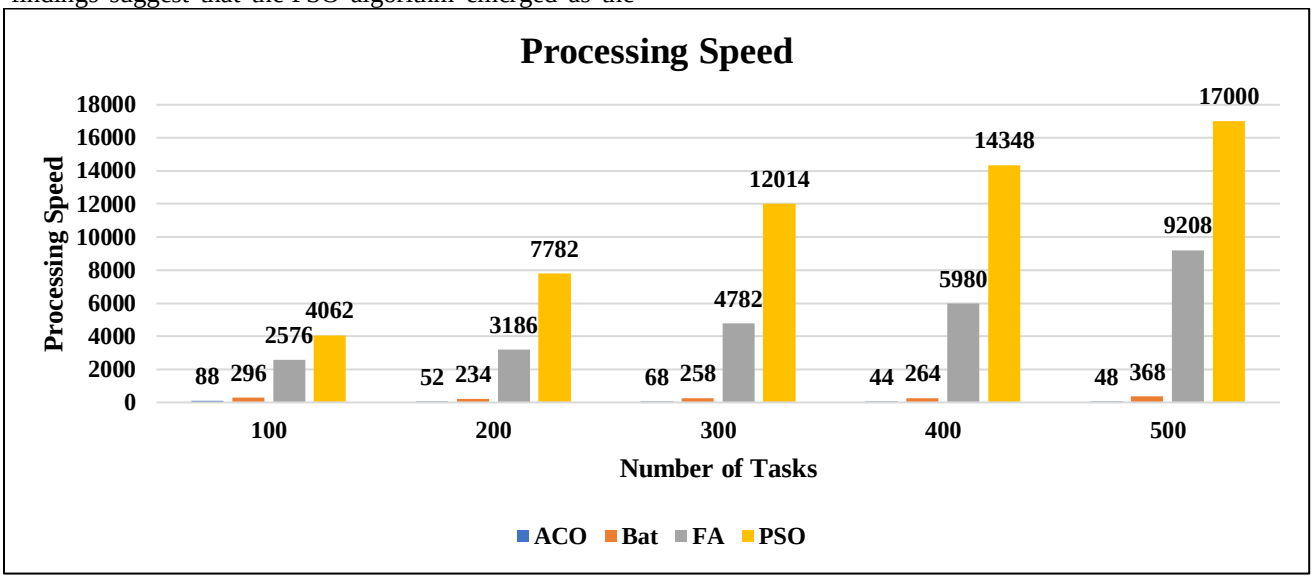

Figure 17 Processing Speed (ms)

In addition to the six metrics examined, our framework incorporates four complementary indicators; Fault Tolerance, as shown in Equation (7); Scalability, as shown in Equation (8); Migration Time, as shown in Equation (9); and SLA Violation Rate, as shown in Equation (10). Collectively, these metrics move the evaluation beyond raw efficiency toward operational robustness by measuring an algorithm's resilience to failure, scalability under workload growth, task-migration overhead, and adherence to QoS commitments. A systematic evaluation of these metrics is planned for the next phase of this research to complete the multidimensional benchmarking suite.

5. OPEN ISSUES AND FUTURE DIRECTIONS

Although nature-inspired load balancing algorithms offer considerable advantages, they encounter challenges such as computational complexity, the necessity for parameter adjustment, limited theoretical analysis, and adaptation to

REVIEW ARTICLE

dynamic environments. To tackle these obstacles, combining these bio-inspired algorithms with both metaheuristic and non-metaheuristic approaches holds promise for enhancing efficiency in cloud resource management. In forthcoming research, a more effective load balancing approach utilizing these methods shall be developed. This proposed work will be subject to validation against pioneering techniques, employing benchmark datasets. The outcomes of this research hold the potential to deliver substantial benefits to a diverse array of stakeholders. The stakeholders will gain from improved resource utilization, increased performance levels, and better cost-effectiveness within Cloud Computing environments.

6. CONCLUSION

Load balancing is critical in cloud environments, helping to achieve efficient resource use and deliver higher-quality services to users. Nature-inspired methods are increasingly recognized as effective solutions to tackle load balancing issues. This study extensively explored the application of swarm intelligence metaheuristic methods for load distribution in cloud computing. It delved into the taxonomy of load balancing in cloud systems as well as metaheuristic techniques and examined their strengths, weaknesses, and applications. The research reviewed various popular swarm intelligence load balancing methods, including ACO, PSO, FA, and the BAT Algorithm. Simulation results were analyzed for load balancing performance parameters such as response time, makespan time, degree of imbalance, processing speed, resource utilization, and standard deviation. In this experimental setup, the ACO algorithm demonstrated superior performance across most parameters. However, it's essential to recognize that each algorithm has its strengths and limitations. Hence, selecting the most appropriate algorithm depends on specific load balancing needs and the accessibility of cloud resources.

REFERENCES

[1] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," Simul. Model. Pract. Theory, vol. 111, no. May, p. 102353, 2021, doi: 10.1016/j.simpat.2021.102353.

[2] International Data Corporation, "Press releases, IDC," 2025. https://www.idc.com/resource-center/press-releases/ ? regions = & languages=eng.[Accessed: Jun. 04, 2025].

[3] S. Afzal and G. Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification," J. Cloud Comput., vol. 8, no. 1, 2019, doi: 10.1186/s13677-019-0146-7.

[4] P. Li, H. Wang, G. Tian, and Z. Fan, "Towards Sustainable Cloud Computing: Load Balancing with Nature-Inspired Meta-Heuristic Algorithms," Electron., vol. 13, no. 13, 2024, doi: 10.3390/electronics13132578.

[5] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 3, pp. 813–824, 2022, doi: 10.1016/j.jksuci.2019.02.010.

[6] J. Zhou et al., "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," J. Cloud Comput., vol. 12, no. 1, 2023, doi: 10.1186/s13677-023-00453-3.

[7] D. Li, L. Wang, W. Guo, M. Zhang, B. Hu, and Q. Wu, "A particle swarm optimizer with dynamic balance of convergence and diversity for large-scale optimization," Appl. Soft Comput., vol. 132, 2023, doi: 10.1016/j.asoc.2022.109852.

[8] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," J. Netw. Comput. Appl., vol. 128, no. April 2018, pp. 64–77, 2019, doi: 10.1016/j.jnca.2018.12.010.

[9] S. K. Bothra and S. Singhal, "Nature-inspired metaheuristic scheduling algorithms in cloud: A systematic review," Sci. Tech. J. Inf. Technol. Mech. Opt., vol. 21, no. 4, pp. 463–472, 2021, doi: 10.17586/2226-1494-2021-21-4-463-472.

[10] R. Rajeshkannan and M. Aramudhan, "Comparative study of load balancing algorithms in cloud computing environment," Indian J. Sci. Technol., vol. 9, no. 20, 2016, doi: 10.17485/ijst/2016/v9i20/85866.

[11] R. M. Singh, L. K. Awasthi, and G. Sikka, "Towards Metaheuristic Scheduling Techniques in Cloud and Fog: An Extensive Taxonomic Review," ACM Comput. Surv., vol. 55, no. 3, 2022, doi: 10.1145/3494520.

[12] N. K. Rajpoot, P. Singh, and B. Pant, "Nature-Inspired Load Balancing Approach in Cloud Computing Environment for Smart Healthcare," ACM Int. Conf. Proceeding Ser., 2022, doi: 10.1145/3590837.3590943.

[13] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," J. Parallel Distrib. Comput., vol. 183, p. 104766, 2024, doi: 10.1016/j.jpdc.2023.104766.

[14] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," 2010 Int. Conf. Green Comput. Green Comp 2010, pp. 357–364, 2010, doi: 10.1109/GREENCOMP.2010.5598294.

[15] S. S. Sefati, M. Mousavinasab, and R. Zareh Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," J. Supercomput., vol. 78, no. 1, pp. 18–42, 2022, doi: 10.1007/s11227-021-03810-8.

[16] A. Thakur and M. S. Goraya, "RAFL: A hybrid metaheuristic based resource allocation framework for load balancing in cloud computing environment," Simul. Model. Pract. Theory, vol. 116, no. June 2021, p. 102485, 2022, doi: 10.1016/j.simpat.2021.102485.

[17] J. Zhou et al., "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," J. Cloud Comput., vol. 12, no. 1, 2023, doi: 10.1186/s13677-023-00453-3.

[18] S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments," Comput. Oper. Res., vol. 110, pp. 159–187, 2019, doi: 10.1016/j.cor.2019.05.022.

[19] P. Geetha and C. R. R. Robin, "A Comparative-Study of Load-Cloud Balancing A lgorithms in Cloud Environments," in Proc. 2017 Int. Conf. Energy, Commun., Data Analytics Soft Comput. (ICECDS), Chennai, India, Aug. 2017, pp. 806–810, doi: 10.1109/ICECDS.2017.8389549.

[20] J. Ma, Linhua and Xu, Chunshan and Ma, Haoyang and Li, Yujie and Wang, Jiali and Sun, "Effective metaheuristic algorithms for bag-of-tasks scheduling problems under budget constraints on hybrid clouds," J. Circuits, Syst. Comput., vol. 30, no. 05, p. 2150091, 2021, doi: 10.1142/S0218126621500912.

[21] J. Kumar and A. K. Singh, "Performance evaluation of metaheuristics algorithms for workload prediction in cloud environment," Appl. Soft Comput., vol. 113, p. 107895, 2021, doi: 10.1016/j.asoc.2021.107895.

[22] X. Huang, M. Xie, D. An, S. Su, and Z. Zhang, "Task scheduling in cloud computing based on grey wolf optimization with a new encoding mechanism," Parallel Comput., vol. 122, no. July, p. 103111, 2024, doi: 10.1016/j.parco.2024.103111.

[23] V. Joshi, "Load Balancing Algorithms in Cloud Computing Vignesh

REVIEW ARTICLE

Joshi To cite this version : HAL Id : hal-02884073 Load Balancing Algorithms in Cloud Computing," pp. 530–532, 2020.

[24] S. K. Sarma, "Metaheuristic based auto-scaling for microservices in cloud environment: a new container-aware application scheduling," Int. J. Pervasive Comput. Commun., vol. 19, no. 1, pp. 74–96, 2023, doi: 10.1108/IJPCC-12-2020-0213.

[25] A. Ramathilagam and K. Vijayalakshmi, "Workflow scheduling in cloud environment using a novel metaheuristic optimization algorithm," Int. J. Commun. Syst., vol. 34, no. 5, pp. 1–15, 2021, doi: 10.1002/dac.4746.

[26] A. Tandon, A. Nayyar, and S. Patel, A novel hybrid meta heuristic approach for optimization of makespan and handling imbalance in task scheduling in cloud environments. Springer Vienna, 2025.

[27] U. K. Lilhore et al., "A multi-objective approach to load balancing in cloud environments integrating ACO and WWO techniques," Sci. Rep., vol. 15, no. 1, 2025, doi: 10.1038/s41598-025-96364-1.

[28] M. Dorigo, "Optimization, Learning and Natural Algorithms," Ph.D. Thesis, Politec. di Milano, 1992, [Online]. Available: https://cir.nii.ac.jp/crid/1573950400977139328.[Accessed: Mar. 23, 2025].

[29] N. K. Rajpoot, P. Singh, and B. Pant, "Nature-Inspired Load Balancing Algorithms for Resource Allocation in Cloud Computing," Proc. Int. Conf. Comput. Intell. Sustain. Eng. Solut. CISES 2023, pp. 827–832, 2023, doi: 10.1109/CISES58720.2023.10183630.

[30] N. Khan, Shagufta and Sharma, "Effective scheduling algorithm for load balancing (SALB) using ant colony optimization in cloud computing," Int. J. Adv. Res. Comput. Sci. Softw. Eng., vol. 4, no. 2, pp. 966–973, 2014.

[31] V. D. E Gupta, "A Technique Based on Ant Colony Optimization for Load Balancing in Cloud Data Center," 2014 Int. Conf. Inf. Technol., pp. 12–17, 2014, doi: 10.1109/ICIT.2014.54.

[32] Y.-Y. Wen, Wei-Tao and Wang, Chang-Dong and Wu, De-Shen and Xie, "An ACO-Based Scheduling Strategy on Load Balancing in Cloud Computing Environment," 2015 Ninth Int. Conf. Front. Comput. Sci. Technol., pp. 364–369, 2015, doi: 10.1109/FCST.2015.41.

[33] S. Seddigh, Milad and Taheri, Hassan and Sharifian, "Dynamic prediction scheduling for virtual machine placement via ant colony optimization," 2015 Signal Process. Intell. Syst. Conf., pp. 104–108, 2015, doi: 10.1109/SPIS.2015.7422321.

[34] M. Farrag, Aya A Salah and Mahmoud, Safia Abbas and El Sayed, "Intelligent Cloud Algorithms for Load Balancing problems : A Survey," 2015 IEEE Seventh Int. Conf. Intell. Comput. Inf. Syst., pp. 210–216, doi: 10.1109/IntelCIS.2015.7397223.

[35] D. G. N. Kaur, C. Kathuria, "Cloud Task Scheduling Based on Enhanced Meta Heuristic Optimization Technique," J. Comput. Sci. Eng., vol. 5, no. 8, pp. 163–168, 2017, doi: 10.26438/jcse/v5i8.163168.

[36] N. Srivastava, Ankita and Kumar, "An energy efficient robust resource provisioning based on improved PSO-ANN," Int. J. Inf. Technol., vol. 15, no. 1, pp. 107–117, 2023, doi: 10.1007/s41870-022-01148-9.

[37] F. Ramezani and J. Lu, "Task-Based System Load Balancing in Cloud Computing Using Particle Swarm Optimization," pp. 739–754, 2014, doi: 10.1007/s10766-013-0275-4.

[38] D. A. Shafiq, N. Z. Jhanjhi, and A. Abdullah, "Load balancing techniques in cloud computing environment: A review," J. King Saud Univ. - Comput. Inf. Sci., vol. 34, no. 7, pp. 3910–3933, 2022, doi: 10.1016/j.jksuci.2021.02.007.

[39] P. Chitra, S and Madhusudhanan, B and Sakthidharan, GR and Saravanan, "Local minima jump PSO for workflow scheduling in cloud computing environments," in Advances in Computer Science and its Applications: CSA 2013, Springer Berlin Heidelberg, 2014, pp. 1225–1234. doi: 10.1007/978-3-642-41674-3_170.

[40] R. M. R. Ashwin, TS and Domanal, Shridhar G and Guddeti, "A Novel Bio-Inspired Load Balancing of Virtual Machines in Cloud Environment," in International Conference on Cloud Computing in Emerging Markets (CCEM), 2014, pp. 1–4. doi: 10.1109/CCEM.2014.7015477.

[41] Z. Aslanzadeh, Shahrzad and Chaczko, "Load Balancing Optimization in Cloud Computing : Applying Endocrine-Particale Swarm Optimization," in 2015 IEEE International Conference on Electro/Information Technology (EIT), pp. 165–169. doi: 10.1109/EIT.2015.7293424.

[42] J. Pan, Kai and Chen, "Load Balancing in Cloud Computing Environment Based on An Improved Particle Swarm Optimization," in 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2015, pp. 595–598. doi: 10.1109/ICSESS.2015.7339128.

[43] H. Zhu, Yongfei and Zhao, Di and Wang, Wei and He, "A novel load balancing algorithm based on improved particle swarm optimization in cloud computing environment," in Human Centered Computing: Second International Conference, HCC 2016, Colombo, Sri Lanka, January 7-9, 2016, Revised Selected Papers 2, 2016, pp. 634–645. doi: 10.1007/978-3-319-31854-7_57.

[44] Xin-She Yang, "Firefly Algorithms for Multimodal Optimization," in Stochastic Algorithms: Foundations and Applications. SAGA 2009, Springer Berlin Heidelberg, 2009, pp. 169–178. doi: 10.1007/978-3-642-04944-6_14.

[45] M. Nikoo, B. Aminnejad, and A. Lork, "Firefly Algorithm-Based Artificial Neural Network to Predict the Shear Strength in FRP-Reinforced Concrete Beams," Adv. Civ. Eng., vol. 2023, pp. 1687–8086, 2023, doi: 10.1155/2023/4062587.

[46] A. P. Florence and V. Shanthi, "A load balancing model using firefly algorithm in cloud computing," J. Comput. Sci., vol. 10, no. 7, pp. 1156–1165, 2014, doi: 10.3844/jcssp.2014.1156.1165.

[47] R. Naveena, N and Santhosh, "Load balancing of real-time services for cloud computing," Int. J. Infin. Innov. Eng. Technol., vol. 1, pp. 73–76, 2014.

[48] A. Goyal and N. S. Chahal, "Bio inspired approach for load balancing to reduce energy consumption in cloud data center," in International Conference Communication, Control and Intelligent Systems, CCIS 2015, 2016, pp. 406–410. doi: 10.1109/CCIntelS.2015.7437950.

[49] M. Aruna, D. Bhanu, and S. Karthik, "An improved load balanced metaheuristic scheduling in cloud," Cluster Comput., vol. 22, no. s5, pp. 10873–10881, 2019, doi: 10.1007/s10586-017-1213-9.

[50] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in Nature inspired cooperative strategies for optimization (NICSO 2010), Springer Berlin Heidelberg, 2010, pp. 65–74. doi: 10.1007/s10586-017-1213-9.

[51] M. A. Elmagzoub, D. Syed, A. Shaikh, N. Islam, A. Alghamdi, and S. Rizwan, "A survey of swarm intelligence based load balancing techniques in cloud computing environment," Electron., vol. 10, no. 21, 2021, doi: 10.3390/electronics10212718.

[52] A. Gupta and H. S. Bhadauria, "Honey Bee Based Improvised BAT Algorithm for Cloud Task Scheduling," Int. J. Comput. Networks Appl., vol. 10, no. 4, pp. 494–510, 2023, doi: 10.22247/ijcna/2023/223310.

[53] S. Sharma, A. K. Luhach, and S. Sheik Abdhullah, "An Optimal Load Balancing Technique for Cloud Computing Environment using Bat Algorithm," Indian J. Sci. Technol., vol. 9, no. 28, 2016, doi: 10.17485/ijst/2016/v9i28/98384.

[54] M. Dhanalakshmi and Anirban Basu, "Energy Efficient Task Provisioning for Distributed Cloud Networks using Meta Heuristic Multidisciplinary Technique," Int. J. Appl. Eng. Res., vol. 11, no. 9, pp. 6195–6199, 2016.

[55] A. Fahim, Youssef and Rahhali, Hamza and Hanine, Mohamed and Benlahmar, El-Habib and Labriji, El-Houssine and Hanoune, Mostafa and Eddaoui, "Load Balancing in Cloud Computing Using Meta-Heuristic Algorithm," J. Inf. Process. Syst., vol. 14, no. 3, pp. 275–284, 2018, doi: 10.3745/JIPS.01.0028.

[56] S. Raj, Bibhav and Ranjan, Pratyush and Rizvi, Naela and Pranav, Prashant and Paul, "Improvised bat algorithm for load balancing-based task scheduling," in Progress in Intelligent Computing Techniques: Theory, Practice, and Applications: Proceedings of ICACNI 2016, Springer, Singapore, 2018, pp. 521–530. doi: 10.1007/978-981-10-

REVIEW ARTICLE

3373-5_52.
[57] L. M. Ibrahim and I. A. Saleh, "A solution of loading balance in cloud computing using optimization of Bat swarm Algorithm," J. Eng. Sci. Technol., vol. 15, no. 3, pp. 2062–2076, 2020.
[58] A. Sundas, S. Badotra, Y. Alotaibi, S. Alghamdi, and O. I. Khalaf, "Modified bat algorithm for optimal VM s in cloud computing," Comput. Mater. Contin., vol. 72, no. 2, pp. 2877–2894, 2022, doi: 10.32604/cmc.2022.025658.
[59] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," Proc. 2009 Int. Conf. High Perform. Comput. Simulation, HPCS 2009, pp. 1–11, 2009, doi: 10.1109/HPCSIM.2009.5192685.

Authors

Vaishali Mehta is a Research Scholar in the Department of Computer Science and Applications at Panjab University, Chandigarh, India. Her research interests include cloud computing, Metaheuristic Load balancing techniques for efficient resource management, and Machine Learning.

Prof. Anu Gupta holds a Ph.D. from Panjab University, Chandigarh, and is currently a Professor in the Department of Computer Science and Applications at Panjab University, Chandigarh, India. Her areas of expertise include Java programming, software engineering, open-source software, and cloud computing.

How to cite this article:

Vaishali Mehta, Anu Gupta, "A Comprehensive Evaluation of Swarm Intelligence Metaheuristics for Efficient Load Balancing in Cloud Computing", International Journal of Computer Networks and Applications (IJCNA), 12(4), PP: 592-613, 2025, DOI: 10.22247/ijcna/2025/36.