Received 9 November 2023, accepted 24 November 2023, date of publication 27 November 2023, date of current version 4 December 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3337146*

SURVEY

# Load Balancing in Cloud Environment: A State-of-the-Art Review

**YOGESH LOHUMI[1], DURGAPRASAD GANGODKAR[1], (Member, IEEE), PRAKASH SRIVASTAVA[1], (Member, IEEE), MOHAMMAD ZUBAIR KHAN[2], (Senior Member, IEEE), ABDULRAHMAN ALAHMADI[2], AND AHMED H. ALAHMADI[2]**

[1]Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun 248002, India
[2]Department of Computer Science, Taibah University, Madinah 42353, Saudi Arabia

Corresponding authors: Prakash Srivastava (prakash2418@gmail.com) and Mohammad Zubair Khan (zubair.762001@gmail.com)

This work was supported by the Deputyship for Research and Innovation, Ministry of Education in Saudi Arabia for under Project 445-9-699.

**ABSTRACT** IT services and resources on-demand through Internetwork are offered by Cloud Computing (CC), including the pay-for-you-go aspect. A lot is offered by the CC paradigm, such as Infrastructure related services, computing services, storage, and environments for deployment are also provided. The objective of this study is to survey one of the significant challenges in cloud computing, which is a multi-variant, multi-constraint issue termed Load unbalancing, resulting in the demising of the scalability, efficiency, and performance of the system. Equilibrium in the server workload distribution is still strived for by cloud service providers. The unbalancing issue is resolved by load balancing solutions in two ways: overloading and underloading. An extensive structural literature analysis of Load balancing and its constituent domains with the inclusion of various parameters, such as scalability, make-span, and throughput, are depicted in this research paper to enhance the QoS. A detailed and organized taxonomy of all the Load balancing algorithms based on nature system state, techniques, functionality, and types is also presented. The major focus of the survey is around the Static, dynamic, hybrid, and nature-inspired Load-Balancing algorithms.

**INDEX TERMS** Virtual-machine, cloud computing, migration, scheduling, ACO, PSO, load-balancing.

## I. INTRODUCTION

Innovation for cloud computing is critically facilitated by virtualization. Hardware and software approaches that allow physical systems to be partitioned among numerous virtual instances that function simultaneously and share the base physical or bare metal resources and equipment draw attention, as noted [3]. The broad scope of research into all key domains and enterprise applications is enabled by the fusion of Cloud computing and Virtualization technologies, as observed in various references [1], [2], [4]. Uniformity is ensured by the ecosystem brace offered by several cloud providers, pointing to the expansion of the business at an affordable price, and automation workflow is properly aligned with legitimate user needs. The allocation of adequate resources to cloud services is a difficult task that depends on the Quality of service (QoS) needs of cloud-based services.

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

In this context, diversity, unpredictability, and resource dissipation lead to issues in resource allocation that are not easily handled by conventional provisioning of resources. The community of software developers has been motivated by the unexpected spike in demand for such systems from customers to design and execute scalable apps in the form of cloud services. The architecture made available by software providers significantly relies on the installed programmes. Due to the limits imposed on the public on-premises resource pools, several app development businesses have migrated the programs to third-party CSP settings. The greater degree of performance, uptime, and the necessary growth of the applications are required from the owners of data centers or cloud service providers. Balancing the load in public clouds spreads workload and resources in a fashion where jobs are assigned to all processing units, allowing for the most effective usage, including all computational resources. More specific Load Balancing (LB) in cloud computing broadly is needed, as indicated [6], [7]. The need for LB in

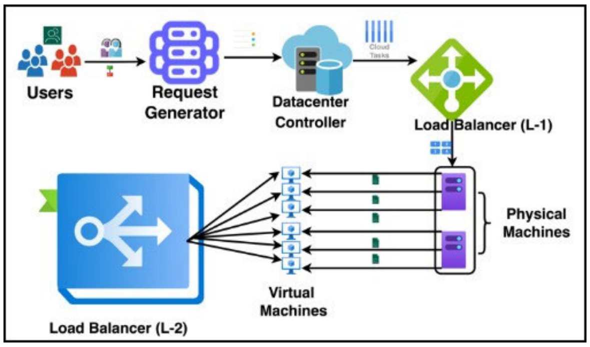

FIGURE 1. General LB mechanism.

cloud computing arises from the versatile nature of the user requests, uncertain and non-probabilistic network flow to a cloud provider, absence of an efficient resource assigner for the user requests, blurry distribution of tasks across computing resources, including their dependability, and varying demands of the resources to the user requests are the critical points for the same [82].

LB distributes the load throughout the system to several nodes. The primary purpose of the algorithms is to pick every job such that it contributes to minimizing the time of execution and resource utilization in cloud data centers. The urgent demands of cloud computing include a challenge and introduce a vast domain for study and development, including all elements of computing resource virtualization. As CC is a market-oriented facility, effective resource scheduling allows legitimate users to focus on their various enterprises to maximize profitability and ROI.

The mechanism of balancing Load in the Cloud is generally a two-phase mechanism, one at the level of physical machines where the load balancer manages a load of physical machines and distributes the load among the associated VMs with every physical device (where task migration is done by two ways Inter Virtual Machine and Intra Virtual Machine task migration) and second at the level of virtual machines where the load balancer manages and balance the load across all the virtual machines through various LB algorithms [98]. The user requests are generated by request generators which are the user jobs that need multiple resources to complete the execution process. DCcontroller does the task management as depicted in Fig 1.

This paper is divided into four sections: Section II provides the literature review. Section III provides discussion and open research directions. Section IV concludes the paper with future scope.

## II. LITERATURE REVIEW

This section presents the literature review of this paper. Firstly, a review of the recent literature on LB where proposed algorithms by researchers are explained and analysed and the concept of LB is explained highlighting its taxonomies, metrics, and existing common algorithms.

Rajgopal et al. [76] proposed a centralized manager methodology suitable for multiple hosts with varying complex practices; this method leads to bottlenecks as immense coordination of operation is required. Ahmed and Khan [81] also proposed a centralized load-balancing strategy for virtual machines, which aggravates the entire performance of the whole process. The fault Tolerance parameter is not considered, which may resist the robustness of the entire system.

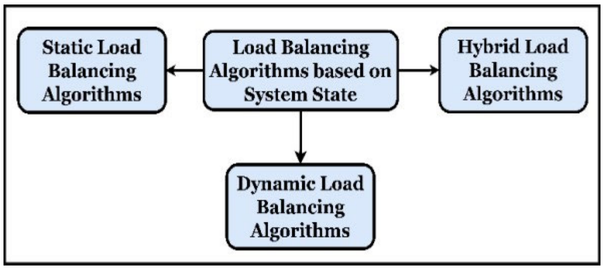
Static Load Balancing Algorithms
Load Balancing Algorithms based on System State
Hybrid Load Balancing Algorithms
Dynamic Load Balancing Algorithms

FIGURE 2. LB algorithms based on the system's state [98].

### A. TAXANOMY OF LB ALGORITHMS

Joshi et al. [77] LB based on Round Robin, which researchers highly use, is easy to implement and performs better when the number of processors is less than the number of operations. Crash and overload of the system can occur when the server requires multiple inputs. Feng et al. [78] introduced an Opportunistic load-balancing algorithm which easily handles incomplete requests in chronological sequence to the current host. The requests are structured steadily as it does not consider the ongoing execution of the host. Gopinath et al. [79] improvised Min-Min and Max-Min LB algorithms which are fast and less complex and improve average makespan of the system. The performance of the Max-Min algorithm is better than Min-Min as it is generally concluded that huge tasks are more in number than smaller tasks, but these strategies can lead and suffer from starvation. Kaur and Mahajan [80] introduced a dynamic and equally spread current execution algorithm which enhances the loading and response time of the data centers but may lead to an increase in cost. Table. 3 shows comparative analysis of various research proposals of various LB algorithms. Throttled and Joined Idle Queue LB techniques improve resource usage by decreasing mean execution time. These approaches are not considered appropriate in specific workload scenarios, as they are only applicable for homogenous systems, need to recognise deadlines, and are complex in nature. [81], [83], [84].

Based on the system's state, LB algorithms initially are of three types viz Static, Dynamic and Hybrid LB Algorithms as illustrated in Fig. 2. Static LB algorithms (SLB) algorithms distribute the receiving workload on the VM server using previous knowledge of the existing servers in the distributed network. These load-balancing techniques contain a pre-defined load schedule specifying the number of tasks distributed on other servers. Dynamic LB is a more adaptable load-balancing approach that can dynamically determine how much load needs to be released during runtime and which machine should be assigned the load.

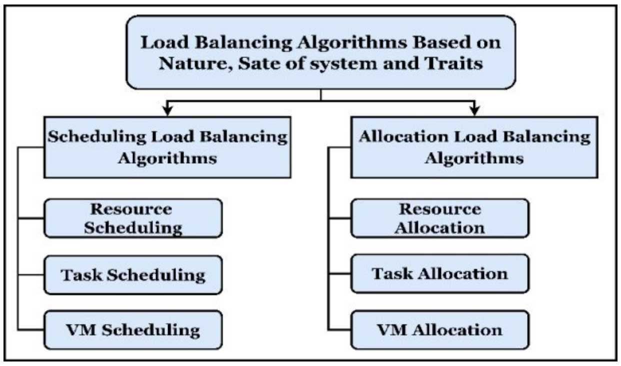
Load Balancing Algorithms Based on Nature, Sate of system and Traits
Scheduling Load Balancing Algorithms
Allocation Load Balancing Algorithms
Resource Scheduling
Task Scheduling
VM Scheduling
Resource Allocation
Task Allocation
VM Allocation

**FIGURE 3.** LB algorithms based on Nature, state of the system and the traits used [98].

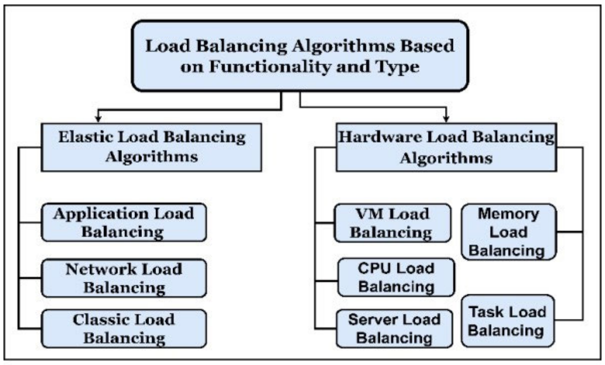
Load Balancing Algorithms Based on Functionality and Type
Elastic Load Balancing Algorithms
Hardware Load Balancing Algorithms
Application Load Balancing
Network Load Balancing
Classic Load Balancing
VM Load Balancing
Memory Load Balancing
CPU Load Balancing
Server Load Balancing
Task Load Balancing

**FIGURE 4.** LB algorithms based on functionality and type [98].

Scheduling and Allocation techniques are crucial in managing the resources, performance monitoring, and positively influencing QoS delivery to legitimate users. Based on the nature and traits used, the load-balancing algorithms are classified as Allocation algorithms and Scheduling Algorithms as depicted in Fig. 3

Allocation algorithms are further classified as task allocation algorithms, Resource allocation algorithms and VM allocation algorithms. At the same time, Scheduling algorithms can be further grouped into three categories: Task scheduling, VM Scheduling and Resource Scheduling algorithms.

Based on the Functionality and type, LB algorithms are classified into hardware Load Balancing (HLB) and Elastic Load Balancing (ELB). HLB algorithms works at infrastructure level. These algorithms primarily manage and distributes the load at physical layer i.e. CPU, memory and storage. HLB algorithms are categorized into five classes i.e. VM, Memory, CPU, Server and Task LB algorithms. ELB algorithms are associated with performance i.e. reliability, scalability, auto-scaling and security related features. Application LB algorithms works at application layer viz. https and http traffic. These algorithms also ensure the security of the clients application. Elastic load-balancing algorithms are grouped into three more types, i.e. Classic, Network and Application LB Algorithms as illustrated in Fig. 4

Based on the techniques employed in LB, The LB algorithms are grouped as Optimized-based and Heuristicbased LB algorithms. Heuristic-based load-balancing algorithms are classified as nature-inspired and classical load-balancing algorithms. Heuristic are proved better when the solutions are needed immediately, these algorithms helps to achieve a decent solution, whereas optimization based algorithms are used to find the most optimal solution of a problem i.e. They help to reach the global [98].Different Nature-Inspired [17] load-balancing algorithms can be categorized in two ways: swarm based load load-balancing algorithms and evolutionary based LB algorithms Swarmbased LB algorithms are further decomposed to ten types: particle swarm optimization (PSO) algorithms [54], [55], [56], [57], ant colony optimization (ACO) [29], [30], [31], [32], [33], [34] algorithms, artificial bee colony algorithms, bees algorithms [74], cuckoo algorithms [100], differential Evolution algorithms, flower algorithms, and bat algorithms. Similarly, Evolutionary based algorithms are grouped as genetic algorithm [14], genetic programming, evolutionary programming, Neuro Evolution, learning classifier system [28], gene expression programming, differential evolution, and firefly algorithms [5] as illustrated in Fig 5.

### B. METRICES FOR LB IN CLOUD COMPUTING

These are the parameters which determine the efficiency of algorithms also called performance of the algorithm. Table 4 shows parameters employed in existing research proposals. Various LB metrics (as demonstrated in Fig. 6) based on the literature review, are being identified by various researchers are explained below

- Response Time (RT) - Mathematically, it can be stated as the delay between the arrival time and the time when the process initially obtains the CPU. Theoretically, it is the time needed to give response to the user's request. The minimum is the RT, the efficient is the LB algorithm [73].
- Throughput (TP) – It is an indicator of the number of jobs and requests appropriately completed and handled in the Virtual machine in unit time interval. It refers to the volume of data transmitted across one node to another. Maximum is the throughput, more efficient is the LB algorithm. [21], [64].
- Scalability - Sudden variations in the number of user requests and the computational load, should not affect the system's performance. The algorithm must be highly scalable for efficient LB [5], [62].
- Resource Utilization - The proportion to which system resources such as storage, processing power, databases, networking, and intelligence are used. Resource usage determines energy consumption in data centres. The maximum utilization of resources optimizes the performance of LB [65], [70].
- Makespan (MS) - Maximum time needed to handle and execute all user requests and assign them to system users. MS is a critical parameter in cloud architecture for scheduling processes [64], [67], [68], [69].
- Associated Overhead (AO) - The LB algorithm needs processing time, memory, and power. All these

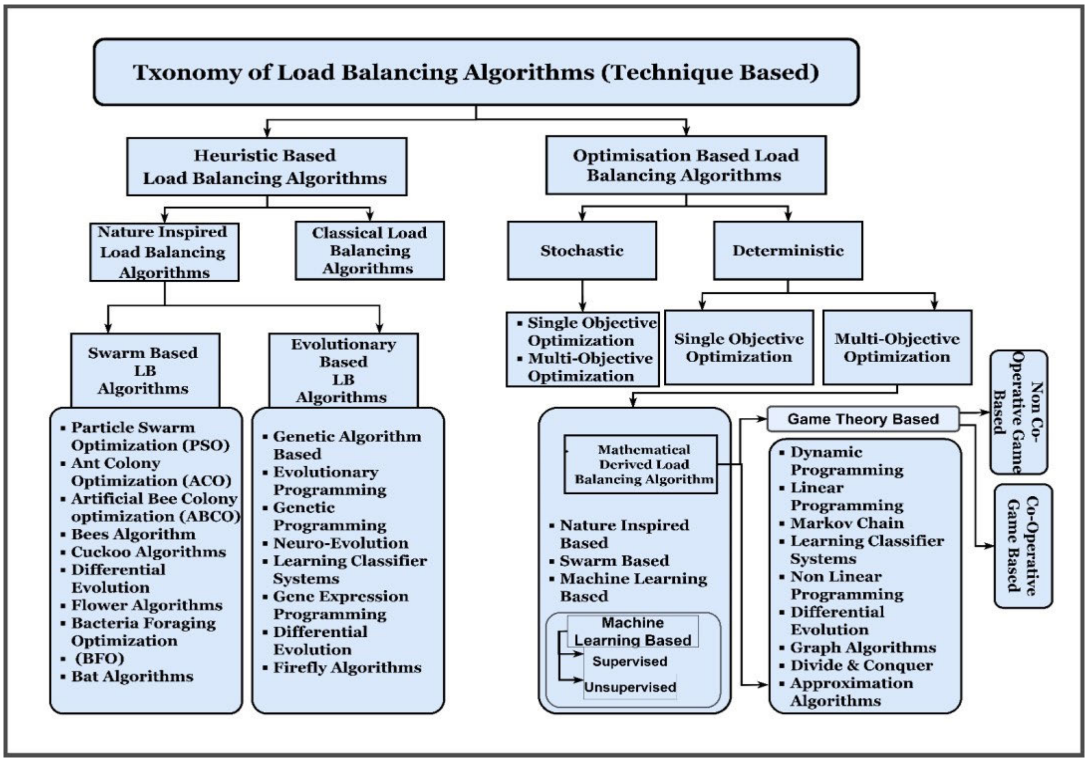
Txonomy of Load Balancing Algorithms (Technique Based)
Heuristic Based Load Balancing Algorithms
Optimisation Based Load Balancing Algorithms
Nature Inspired Load Balancing Algorithms
Classical Load Balancing Algorithms
Stochastic
Deterministic
Swarm Based LB Algorithms
Evolutionary Based LB Algorithms
▪ Single Objective Optimization
▪ Multi-Objective Optimization
Single Objective Optimization
Multi-Objective Optimization
▪ Particle Swarm Optimization (PSO)
▪ Ant Colony Optimization (ACO)
▪ Artificial Bee Colony optimization (ABCO)
▪ Bees Algorithm
▪ Cuckoo Algorithms
▪ Differential Evolution
▪ Flower Algorithms
▪ Bacteria Foraging Optimization
▪ (BFO)
▪ Bat Algorithms
▪ Genetic Algorithm Based
▪ Evolutionary Programming
▪ Genetic Programming
▪ Neuro-Evolution
▪ Learning Classifier Systems
▪ Gene Expression Programming
▪ Differential Evolution
▪ Firefly Algorithms
Mathematical Derived Load Balancing Algorithm
▪ Nature Inspired Based
▪ Swarm Based
▪ Machine Learning Based
Machine Learning Based
Supervised
Unsupervised
Game Theory Based
▪ Dynamic Programming
▪ Linear Programming
▪ Markov Chain
▪ Learning Classifier Systems
▪ Non Linear Programming
▪ Differential Evolution
▪ Graph Algorithms
▪ Divide & Conquer
▪ Approximation Algorithms
Non Co-Operative Game Based
Co-Operative Game Based

**FIGURE 5.** LB algorithms based on techniques [98].

parameters, including inter-process communication and migration of tasks, probably add up for associated overhead. The minimum AO, more efficient is the LB [59], [71], [72].
- SLA violations - Counts the set of SLA violation elements that have been reduced in terms of timeline limitation, prioritization, and. SLA

### C. STATIC LB ALGORITHM
Static LB techniques do not consider the system's condition or metrics, such as processing while distributing requests [6]. These algorithms distribute requests evenly across VMs or according to other principles not influenced by constraints [8], thus ideal for systems with hardly any change in load. These algorithms need a thorough understanding of server facilities to improve system throughput [8]. Kumar et al. [75] employed sub-optimal and optimal techniques in which resource descriptions of structured techniques are aggregated, and jobs are assigned to load balancers; if the load balancer does not make a precise judgement, suboptimal components are selected. The drawbacks of these algorithms are that they do not have scope for live changes in user requests, have low throughput and fault tolerance. To overcome these drawbacks, dynamic LB algorithms are taken into account.

### D. DYNAMIC LB ALGORITHM
The cloud provider deploys applications in a dynamic environment. For DLB algorithms, various research challenges must be considered, such as how often resource scheduling must be called, which node leads the LB, obtaining VM load information, and load migration across nodes. Many potential solutions to load unbalance have already been presented by various researchers. A variety of services and resources are adaptable in live production environments therefore CSP cannot rely upon this circumstance. On existing information, while considering analytics in real time, the clients' criteria are provided for adaptability and flexibility [9], [10], [11]. The algorithms developed to accomplish dynamic LB, the ecosystem is highly adaptive to variations in requests over time. It is challenging to imitate a production environment [12], even though it is possible. Zhao et al. [18] introduced a dynamic bin packing strategy and divide and conquer methodology

General overview of static and dynamic LB algorithms is illustrated in Table 1

### E. NATURE INSPIRED ALGORITHMS
Nature-inspired algorithms are meta-heuristic algorithms that are influenced by or mimic biological activities [17] already described by physical sciences. General overview of nature inspired LB algorithms is illustrated in Table 2.

#### 1) ANT COLONY OPTIMIZATION
Ant colony optimization (ACO) is a meta-heuristic technique for finding optimized solutions to multimodal optimization problems. It is appropriate for challenges with a broad and complicated outcome space. It is based on the food-gathering patterns of ant colonies [29]. Fig.7 depicts the process

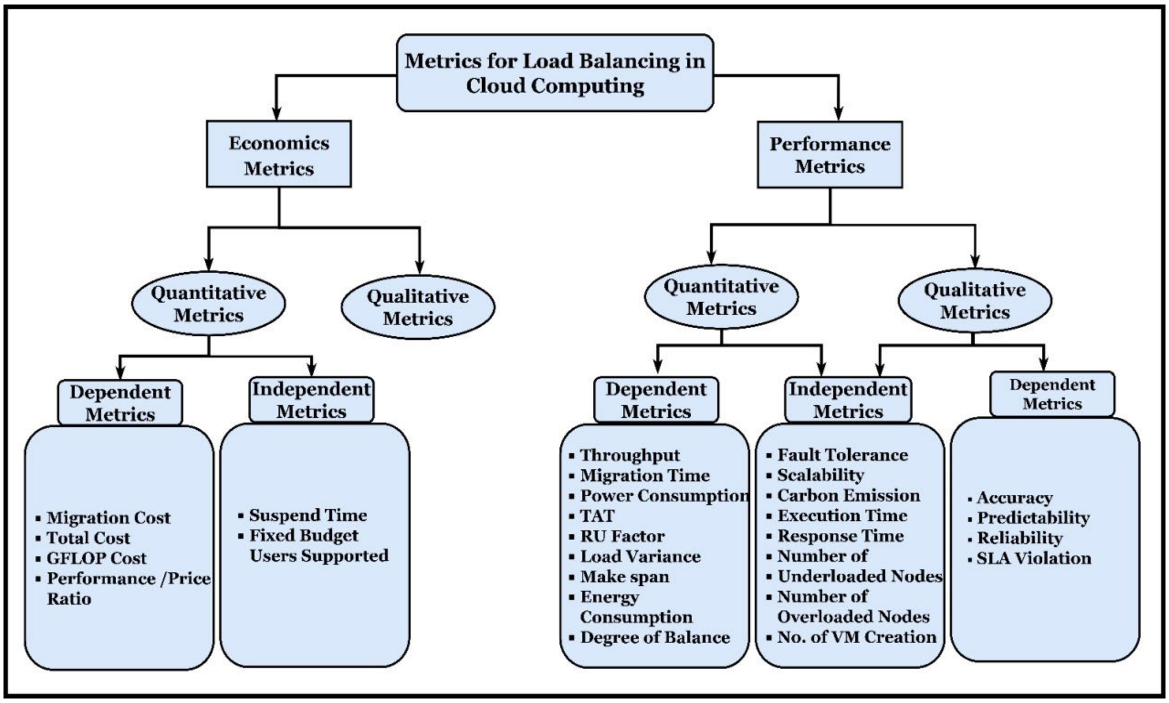

**FIGURE 6.** Taxonomy for LB metrics [98].

**TABLE 1.** General overview of static and dynamic LB algorithms.

| Reference | Approach | Brief description |
|---|---|---|
| [76],[8] | Static LB | Distribute requests evenly among VMs or according to other non-constrained principles [8]. Sub-optimal and optimal strategies where structured technique's resource description is cumulated, and tasks are sent to the load balancers [76]. |
| [92],[93] | Dynamic LB | Provide flexibility in selection of VMs and service user requests, minimize resource Utilization [92]. In case of computational load variation high latency can be seen [93] |
| [94],[95] | Dynamic LB | Maximized the percentage of usage of resources, priority is not included in tasks[94], for large scale systems efficient distribution mechanism of load, congestion can occur[95] |

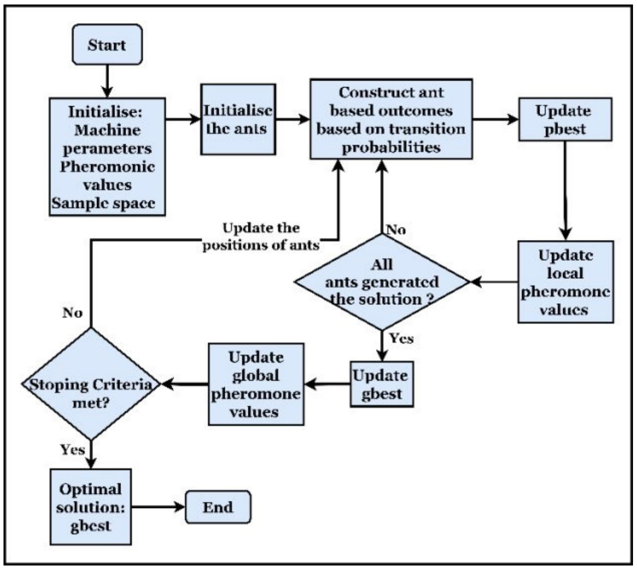
Start
Initialise: Machine perameters Pheromonic values Sample space
Initialise the ants
Construct ant based outcomes based on transition probabilities
Update pbest
Update local pheromone values
All ants generated the solution ?
No
Yes
Update the positions of ants
Update gbest
Update global pheromone values
Stoping Criteria met?
No
Yes
Optimal solution: gbest
End

**FIGURE 7.** Flowchart of ACO.

followed in ACO with the help of a flowchart. For LB in cloud computing, Verma et. al [31] proposed an enhanced LB approach to enhance the Quality of Service and optimized resource allocation. The entire process is done in two phases: Generation of Ants; Ants are created after frequently monitoring the system for overburdened or noload nodes and to locate the candidate node; as per the exploring guidelines, the ant is hunting for candidate nodes in its ecosystem that match the LB parameters. Precise placement of virtual machines [32], [33] in cloud LB is a crucial phase. For efficient VM placement Xing et. al. [34] introduced a methodology named ETA-ACO in which Virtual-Machine Placement is done in descending order according to the clients requirements. This Technique mounts freshly generated solutions over already generated grouped solutions by distributing the elements of the optimum solutions.

Muteeh et al. [30] introduced the multiple resource LB algorithm(MrLBA) for workflow scheduling based on Ant colony Optimisation(ACO), which aims to complete the whole workflow execution by taking multiple metrics into account, primarily cost and makespan, also Quality of service (QoS), execution cost, execution time, and user timelines. This model consists of crucial components explaining various processes' interconnection. Initially, parsing of input

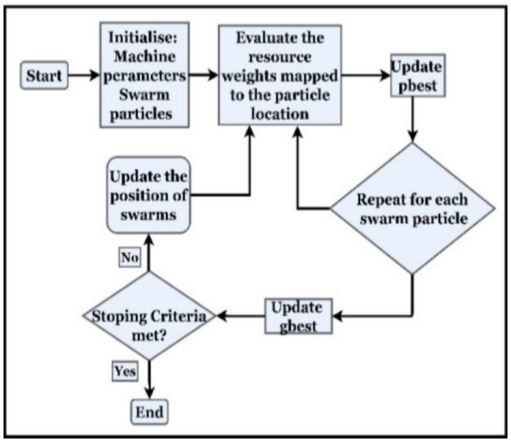
Start
Initialise: Machine perameters Swarm particles
Evaluate the resource weights mapped to the particle location
Update pbest
Repeat for each swarm particle
Update the position of swarms
Update gbest
Stoping Criteria met?
No
Yes
End

FIGURE 8. Flowchart of PSO.

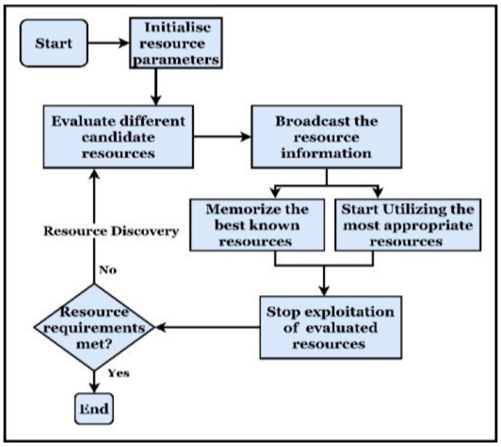
Start
Initialise resource parameters
Evaluate different candidate resources
Broadcast the resource information
Memorize the best known resources
Start Utilizing the most appropriate resources
Resource Discovery
No
Resource requirements met?
Stop exploitation of evaluated resources
Yes
End

FIGURE 9. Flowchart of ABC.

workflow is done. Workflow jobs are organized based on the total count of offspring, execution duration, and size of data. The jobs are grouped based on the number of offspring, significant depth, processing time, and size of the file.

### 2) PARTICLE SWARM OPTIMIZATION

PSO is a stochastic optimization technique that mimics social demeanours such as bird herding or fish pedagogy. In a PSO system, a swarm of components (called particles) meander in state space [54]. Fig. 8 shows the Flowchart of PSO. Alsaidy et al. [42] proposed heuristic-based improved initialization of PSO for task scheduling in CC [95]. Longest job allocation to quickest processors and minimal competition time approaches are employed to initialize PSO. LJFT-based PSO and MCT-based PSO initialization methodologies are examined for the degree of imbalance, total energy consumption execution time and make span.

Saleh et al. [55] also presented improved particle swarm optimization for enriching resource usage and efficient task allocation in cloud environments, comprising batch creation, resulting in minimizing makespan by 50%. The proposed Algorithm IPSO is more promising for a large number of tasks.

Alguliyev et al. [56] presented task migration aware LB, where tasks are migrated from overwhelmed virtual machines to underloaded virtual machines, which results in equalized distribution of load among all virtual machines. This methodology leads to minimizing the task execution and task migration time.

Kumar and Sharma [57] introduced a task processing framework, and the suggested method generates the saccadic group with simulated values. It enhances many relevant metrics, such as the user's request acceptance rate, completion time, throughput and cost under a sequence of evaluations on different data sources. Service level agreement violation parameters should be considered while developing the PSO -BOOST framework and have more completion time for dynamic load.

The Cloud RAN integrates the cloud into the access network topology and provides for increased network capability by virtualizing the base station and pooling cloud resources. The framework provides the BBU Pool. Michel et al. [74] provide a method for LB for a C-BBU RAN's pool. It is focused on the artificial bee colony meta-heuristic optimization (ABC algorithm) [25] as well as the Min-Min Algorithm (LBMM). These strategies are employed to increase network efficiency by ensuring the high reliability of centralized and pooled assets. Compared to RR and MINMIN techniques, it outperforms them both concerning efficiency and execution time. The flow-chart of the ABC algorithm is depicted in Fig. 9

### 3) ARTIFICIAL BEE COLONY ALGORITHM

## *F. HYBRID LB ALGORITHM*

Both static and dynamic load-balancing algorithms have been suggested and executed previously but have not been efficient and effective in LB, which enabled algorithmic hybridization. The qualities of hybrid approaches are inherited from static and dynamic load-balancing techniques, and efforts are made to overcome the drawbacks of both methods.

Different meta-heuristic and heuristic strategies have been presented to handle the scheduling and load-balancing challenges in cloud computing. The scenario grows increasingly difficult due to the number of requests, virtual machines, and infrastructure expansions. Despite the failure of meta-heuristic approaches to produce relatively immediate optimal answers, the PSO algorithm has been found less efficient in achieving the local maxima. Hybrid systems integrate the attributes of several methodologies to get the best solution and have grown in popularity in recent years.

Thakur et al. [5] proposed RAFL, a metaheuristic allocation of resources approach for cloud LB using hybrid optimisation algorithm PPSO-DA (Particle Swarm Optimization - Dragonfly algorithm), and to dynamically reduce the load mismatch between individual operational servers and actual anticipated resource capacity (Processing and Memory Unit ), this keeps active physical devices from being

**TABLE 2.** General overview of nature inspired LB algorithms.

| Method | Advantage | Disadvantages |
|---|---|---|
| Ant colony-based LB algorithm [31],[35],[86] | Efficient LB, | Limited to only simulation, not yet implemented in real time cloud environment |
| | Fused efficiently with other heuristic based algorithms Minimize processing time | |
| Artificial Bee Colony based LB Algorithms [75][87] | Minimizes the time of task migration and execution time. | Reduces the resource usage, Less scalable |
| PSO based LB algorithms [43],[56],[57] | Reduces the response time and the number of task migration | High complexity and low scalability |
| Honeybee Foraging [88][89][90] | Reduced response and makespan time, enhances the complexity of the system | When the number of VM's increases or decreases throughput remains unchanged. |
| LB algorithm based on osmosis [18][91] | Tasks can be reallocated easily, Supports homogeneous and heterogeneous setting mode in the virtual machines. | The algorithm is not centralized and cannot possess parallel execution of tasks |

overloaded or underloaded and ensures that their capacity of resources is employed in a structured way. Scheduling is a fundamental concern. Kumar et al. [21] proposes a hybrid task scheduling system. GA and PSO are integrated to assign adequate resources to user requests. In contrast to GA and PSO techniques, the HGPSO reduces execution time while increasing availability and scalability. User requests are stored in a queue manager, and the manager's results are assigned to the hybridization of PSO and GA.

Ragmani et al. [13], [15] proposed a combined approach composed of two techniques: ant colony optimization and fuzzy logic for the improvement of LB in cloud environments. The proposed hybrid approach has taken LB and response into account. Pang et al. [14] proposed the estimation distribution algorithm(EDA) - genetic algorithm (GA), which has a fast convergence rate, and robust searching ability, which minimized the request completion time. Velliangiri et al. [16]. Suggested HESGA technique has merits of both genetic and electrical search algorithms. The GA gives the locally optimized outcome, while the Electrical search method delivers the best global optimum. Manikandan et al. [20] proposed a hybrid algorithm HWOAmBA based on bee optimisation for maximizing resource utilization and having a comparatively faster convergence rate, lesser execution time and makespan. Princess et al. [22] integrated harries hawks optimization and pigeon inspired optimization algorithms to balance the load optimally between the virtual machines and adequate resource utilization including tasks response time. The major limitations of VMs are booting time and consumption of unnecessary resources, to overcome the problem Manikandan et al. [23] introduced BWFSO which is the hybridisation of fuzzy C-means clustering, black widow optimisation and FSO for user requests scheduling, allocation of resources and to maximize resource utilization.Wesabi et al. [24] introduces novel hybrid metaheuristic algorithms for energy-efficient resource allocation (HMEERA) in the CC context. The suggested model primarily does the extraction of features based on enduser requests, followed by feature reduction by PCA and subsequently uses the combined characteristics to optimize the allocation of resources. This merging of GTOA and RSO algorithms helps to enhance resource allocation across VMs in cloud data centers. Kruekaew and Kimpan [25] proposed MOABCQ algorithm is indeed a user job requests scheduling approach in cloud computing which is an integration of Artificial Bee colony and Q-learning approach, which is a Machine learning method that enables the ABC approach to work faster. The suggested solution seeks to improve scheduling and resource usage, maximize VM throughput, and provide balancing of workload across VMs based on metrics such as cost, makespan time and resource consumption, all constraints of ongoing concerns. Adequate resource utilization is obtained when an appropriate LB is accomplished in the cloud. Jena et al. [28] also integrated an improved Q-learning approach with Modified particle swarm algorithm, This integration mechanism is employed to modify the MPSO's frequency using the pbest and gbest based on the objective expected by the improved Q-learning.

The suggested approach equalizes the load by reallocating it to the suitable VMs based on their fitness values. When the comparison is made independently with modified particle swarm optimization and Q-learning, the suggested technique enhances throughput, makespan, and energy usage while balancing the load and significantly minimizes jobs average waiting time. However, LB and task scheduling are considered NP-hard optimization problems. To overcome this limitation, Neelima and Reddy [26] proposed a task scheduling-based LB algorithm: The adaptive Dragonfly algorithm (ADA), for improving metrics such as execution cost and time. The ADA is a hybridization of nature-inspired firefly and dragonfly algorithms. A multi-objective function is also included based on three parameters: no end-user requests(load), processing costs and the total time of completing all requests.Two distinct balancing approaches are utilized to increase the load on the resources. The first technique may be executed post-job allocation to resources,

**TABLE 3.** General overview of LB algorithms.

| S/N | Proposed Algorithms | Merits | Demerits | Implementation Tools and Simulation Environment |
|---|---|---|---|---|
| 1 | Manikndan et al. [21] | Minimize the computation cost and execution time. | Less efficient for peak demands. | CloudSim |
| 2 | O. Y Abdulhammd. [36] | Reduce the makespan time, Processing time and degree of imbalance | Increased searching time for Candidate VM.ie local optima not considered. | CloudSim |
| 3 | Jangra & Mangla. [39] | Less makespan time and latency time. Energy efficient. | Algorithm Failure when addition of new nodes to DC, change in workload | Matlab |
| 4 | Shafiq et al.[42] | Reduce Makespan and Efficient resource utilization. | Migration count can be considered for better performance. | CloudSim |
| 5 | Alsaidy et al.[43] | Reduce degree of imbalance, makespan, total execution time ,and energy consumption. | Creates initial particles which search from a single starting position. | Matlab |
| 6 | Negi et al.[44] | Reduce completion time and makespan time, Minimize Energy consumption. | Not efficient for high storage tasks and Energy efficiency mechanism is missing. , | CloudSim |
| 7 | Mapetu et al.[45] | Less SLA violation, low running time and time complexity. | Does not include a peak demands scenario and is complex to implement in a real time environment. | CloudSim |
| 8 | Semmoud et al.[46] | Reduce migration cost and response time, increase resource utilization. | Does not work efficiently for dependent tasks. | CloudSim |
| 9 | Kaurav and Yadav[47] | Reduce requests competition time, Improved scheduling mechanism for Virtual machines. | Precencedence and chronology of tasks is not taken into account. | CloudAnalyst |
| 10 | Xavier and Annadurai. [38] | Minimize overall makespan for heterogeneous VMs, restrict local convergence, and probe global intelligence searching. | Not compatible with independent tasks . | CloudSim |
| 11 | Devagnanam and Elango. [37] | Enhance CPU and Memory Utilization rate. | Doesn't consider multi-objective resource allocation. | Cloudsim, JAVA |
| 12 | Kaur and Kaur [40] | Reduce Cost and make span | Non-compatible with non- uniform load distribution. | CWS (cloud workflow simulator) |
| 13 | Aliyu and Souley[41] | Reduce cost and use physical memory. i.e. a threshold value is set for each VM. | Not Robust, may behave abruptly in case of any failure. | CloudSim |
| 14 | Alamin et al.[48] | Reduce response time, and less utilized resources are ignored. | Increase in waiting time | CloudAnalyst |
| 15 | Adhikari et al[49] | Reduce response time and degree of imbalance | Performs better only in homogeneous environment, low scalability | CloudSim |
| 16 | Ziyath and kumar[50] | Enhance resource usage, maximized LB | Low fault tolerance and QoS | CloudSim , CloudAnalyst |
| 17 | Vinothini and Balasubraman[51] | Reduce migration time and enhance resource usage, maximized LB | Reduced fault tolerance and scalability and increased energy consumption. | CloudSim |
| 18 | Attiya et al.[52] | Performance and resource application increase | Low fault tolerance and response time. | CloudSim |
| 19 | G and Wu [53] | Increase reliability and scalability | Low fault tolerance, QoS and Increase in waiting time | CloudSim |
| 20 | Milan et al.[54] | High fault tolerance, resource utilization and low overhead | High Makespan and Response time | CloudSim |

whereas the second method can be executed previously. The first situation produced poor outcomes due to excessive transmission within the CPU, high computation costs, and the necessity for extra time to accomplish the job. Consequently, the deadlines for certain activities may expire, and overall system efficiency will begin to deteriorate; for this, Moori et al. [27] presented LATOC, a smart blend of AHP-TOPSIS and optimized Particle Swarm Optimization showed significant improvements in average resource usage and total execution time.

The difficulties associated with processing large amounts of data in edge computing settings because of the rise in connected devices and data production. It focuses on data skewness, or the situation where certain data blocks demand more power from the processor. By classifying data blocks according to their importance, the article suggests a data skew-aware method for assigning data blocks to edge servers or the cloud, lowering cloud processing expenses. Based on the research findings, this strategy can reduce processing costs compared to existing techniques by up to 35%. CPU

**TABLE 4.** Comparative analysis of various research proposals based on LB metrices.

| Work | Scalability | Makespan | Response Time | Execution Time | Execution Cost | Overhead | Resource Utilization | Throughput | Migration Time |
|---|---|---|---|---|---|---|---|---|---|
| [5] | ✓ | ✓ | × | × | × | × | ✓ | × | × |
| [21] | × | ✓ | × | ✓ | × | × | ✓ | × | × |
| [22] | ✓ | × | × | × | × | × | × | ✓ | × |
| [12] | × | ✓ | × | × | ✓ | × | ✓ | × | × |
| [59] | ✓ | ✓ | × | × | × | × | × | × | × |
| [60] | ✓ | × | × | × | × | ✓ | ✓ | × | × |
| [13] | × | ✓ | × | ✓ | × | × | ✓ | × | × |
| [61] | ✓ | × | × | × | × | × | × | × | × |
| [62] | × | ✓ | × | ✓ | × | × | ✓ | × | × |
| [63] | ✓ | × | ✓ | × | ✓ | × | × | × | × |
| [64] | × | × | ✓ | ✓ | × | × | × | × | × |
| [65] | × | ✓ | × | ✓ | ✓ | × | × | ✓ | × |
| [66] | × | × | × | ✓ | × | × | × | × | ✓ |
| [67] | × | × | × | × | × | × | ✓ | × | × |
| [68] | × | ✓ | ✓ | ✓ | × | × | × | × | × |
| [69] | × | ✓ | × | ✓ | × | × | × | × | × |
| [70] | × | ✓ | × | ✓ | × | × | × | × | × |
| [71] | × | × | ✓ | × | × | × | ✓ | ✓ | × |
| [72] | ✓ | ✓ | × | × | × | ✓ | × | × | ✓ |
| [73] | ✓ | × | × | ✓ | ✓ | ✓ | × | ✓ | × |
| [74] | ✓ | × | ✓ | ✓ | ✓ | × | × | × | × |

utilisation control in Big Data processing impacts energy usage. DVFS is used for processing, sample to estimate resource requirements, and partition data into uniform blocks. Our strategy works better than variety-oblivious approaches, according to the results, especially when there are short deadlines and we can make better use of DVFS. Future initiatives include investigating renewable energy sources for efficiency gains and optimising data processing by taking energy costs across data segments and locations into account [102]. In [103] introduced a significance-aware approach for enhancing QoR in budget-constrained big data processing. It efficiently allocates resources, outperforming existing methods in benchmarks. Advantages of SAIR include strong performance, low overhead, and simplicity, while Approx-Hadoop excels with uniform data but falters with uneven distribution, incurring high overhead. Processing costs in cloud computing are heavily impacted by resource allocation, particularly in big data applications. Variations in VM performance may arise from a number of data. Data-variety-aware allocation of resources strategy that divides data into blocks and evaluates each block's importance to choose the right virtual machines (VMs) in order to minimise costs. For accumulative applications, we reduce costs by employing targeted sampling techniques [104]. Gapprox reduces the processing of data while preserving an acceptable level of result quality by using sampling clusters to increase accuracy. Groups of the provided data are separated, taking intra- and inter-cluster variation into account, and block and sample sizes are selected to guarantee confidence and error boundaries. According to data from experiments, where a 5% mistake is acceptable with 95% certainty, Gapprox beats the most recent methods, improving processing time by as much as 17× when compared to ApproxHadoop and 8× when compared to Sapprox [105]. Warehouse Scale Computers (WSC) are widely used to handle data from several sources for a variety of big data jobs. Understanding that the importance of the data varies, we give priority to and more resources are allocated to significant data segments, improving the time and cost-effectiveness of WSC. We provide a low-overhead technique for determining the relevance of data portions, which results in a 24% time and 9% cost improvement in resource allocation. Furthermore, this ranking facilitates more effective use of renewable energies in WSCs and expedites the approximation of big data job outputs [106]. Due to cloud cost limits, edge computing configurations require effective big data processing, especially with the increase in connected devices and data volume. One major difficulty is data skewness, which occurs when specific data blocks require greater processing resources. Current techniques for allocating resources ignore this problem. By classifying data blocks according to their importance and assigning less important ones to edge servers, this article suggests a data skew-aware solution that can save cloud processing costs by up to 35% when compared to existing techniques [101]. Big Data's diverse data, which comes from a range of sources and is distributed unevenly, causes large differences in CPU resource usage. Previous study has neglected to address this issue. In order to mitigate this issue, our research utilises Dynamic Voltage and Frequency Scaling (DVFS) to minimise computation energy usage, taking into account two distinct deadline scenarios as limitations. Prior to using DVFS, we estimate the processing time and required frequencies. Our technique

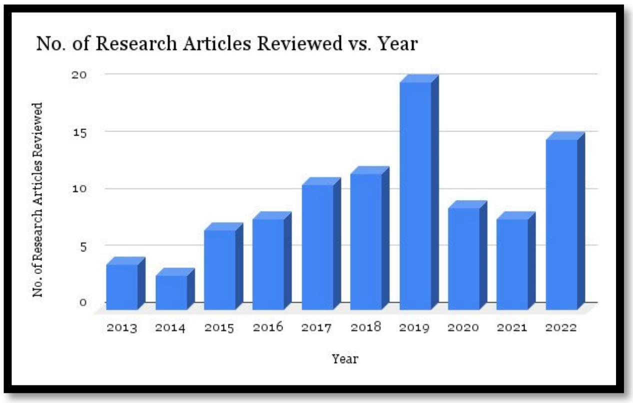

**FIGURE 10.** No of research articles vs year.

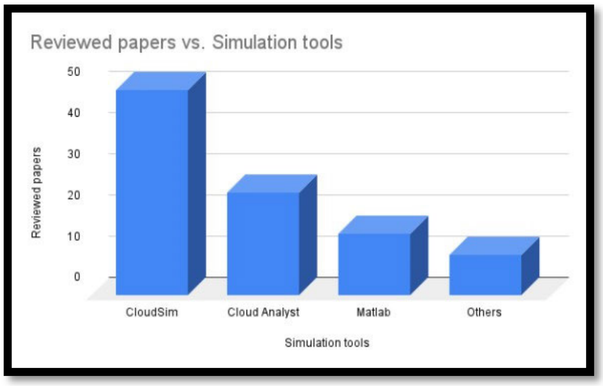

**FIGURE 11.** Simulation tools of the reviewed articles.

achieves up to 15% savings in energy usage using DV-DVFS, which is better compared to different situations, according to the outcomes of experiments with real-world data sets [102]. While big data processing is becoming more and more common in many industries, Quality of Result (QoR) is being challenged by cost concerns. SAIR provides a method of improving QoR for aggregative jobs by giving priority to important data segments while staying within budgetary constraints. The 95% CI with a 5% error margin and statistical techniques are used to determine the most and least significant data segments. With assessments across subdomains such as writings agreements, and logs showing QoR improvement while adhering to budget restrictions, SAIR optimises QoR under desired time frames and budgetary limits [103].

## III. DISCUSSION

The survey majorly levitates around the Static, dynamic, hybrid, and nature-inspired Load -Balancing algorithms. Further, precise future research directions have also been included, which will help researchers find an optimal loadbalancing strategy that could overcome all issues and challenges in existing Load-Balancing algorithms and can be implemented in a real cloud environment. This study comprises research papers in resource scheduling and LB based on a collection of 100 research papers as given in Fig. 10 papers out of a vast array of 900 research papers published in prominent journals, symposiums and conferences.

Based on the analysis, this literature is majorly based on two cloud simulation tools CloudSim and Cloud Analyst, Fig. 11 depicts the percentage distribution of various simulation tools employed.

### A. RESEARCH DIRECTIONS

Based on the analysis, enough unresolved research challenges need to be given more attention. The study revealed possibilities for enhancements in the load-balancing techniques for further investigations, optimizing the Cloud computing services. These are mentioned below:

- Primarily research is based on a simulation environment, plenty of algorithms have been designed to mimic the cloud environment, but it is very challenging to implement them in real-time; still, there are only a few algorithms applicable in a real-time environment. Open-source environments such as OpenStack and Cloud Foundry can be used to implement algorithms prone to real-time challenges [96], [97].
- Complications in the methodology might arise due to various factors, such as the rapid vertical scaling of servers to the cloud Data Centre, a priority job awaiting execution, changes in workflows, and system configurations. A robust mechanism is required to handle the aforementioned issues, improving scalability, performance, and throughput.
- Scheduling aware LB algorithms such as Round Robin results in uneven distribution of the load among the virtual machine servers as it works in cyclic fashion, and VM state are not saved.
- Fusion of two or multiple algorithms increases the complexity of the system, as static LB algorithms can be integrated with Nature-inspired algorithms or other dynamic algorithms so that algorithms can be light and also overcome hybridisation challenges.
- The applications of Cloud computing in the health sector is proved to be an asset for society, and an efficient task allocation algorithm focused on the health sector can be designed so that society can be benefited from it.
- Task scheduling and load unbalancing problems are considered as NP-hard problems. Nature-inspired algorithms have managed to solve it to some extents still there are many open challenges, such as the design of the optimal fitness function, which could evaluate the candidate resources precisely so that scalability can be assured for tasks with dynamic resource requirements.
- QoS-based LB follows a geographical LB mechanism, which delivers an acceptable QoS even in the case of resource failure. Still, it leads to high monitoring overheads which require in-depth analysis of the scenario so that the above challenges can be addressed.
- Most load-balancing methods consider the fault tolerance parameter lightly, which is crucial when proposing a robust technique in real-time cloud environment.

## IV. CONCLUSION AND FUTURE SCOPE

An essential feature in the CC domain, LB is employed to improve workload distribution and resource management, with the aim of minimizing the total response time of the system. Concerns related to LB, such as resource scheduling, task migrations, and resource utilization, have been addressed by many methodologies and approaches. Different methods in the binding domain of LB were explored in this research. In recent years, the challenges associated with LB have been examined by scholars, who have comprehensively evaluated the suggested methods. Although several ways have been presented, various challenges in the cloud environment remain unresolved, including VM migration and fault tolerance concerns. Considerable room is offered to research professionals to design advanced and effective load-balancing methods for cloud environments through this literature survey. This study, encompassing a review of existing and proposed load-balancing strategies, will be helpful for researchers in identifying research challenges connected to LB, particularly in reducing response time and avoiding node breakdowns. In the dynamic landscape of cloud computing, OpenStack presents a platform where open questions remain: How can more robust fault tolerance mechanisms be developed for LB in the cloud within the context of OpenStack? What innovative approaches can be devised to enhance VM migration in a cloud environment, specifically when utilizing cloudfoundry and OpenStack?

## REFERENCES

[1] R. Buyya, "Introduction to the IEEE transactions on cloud computing," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 3–21, Jan. 2013.

[2] L. Moser, B. Thuraisingham, and J. Zhang, "Services in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 172–174, Mar. 2015.

[3] R. Birke, A. Podzimek, L. Y. Chen, and E. Smirni, "Virtualization in the private cloud: State of the practice," *IEEE Trans. Netw. Service Manage.*, vol. 13, no. 3, pp. 608–621, Sep. 2016.

[4] Y. Kotb, I. Al Ridhawi, M. Aloqaily, T. Baker, Y. Jararweh, and H. Tawfik, "Cloud-based multi-agent cooperation for IoT devices using workflow-nets," *J. Grid Comput.*, vol. 17, no. 4, pp. 625–650, Dec. 2019.

[5] 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, Apr. 2022, Art. no. 102485.

[6] K. Mahajan, A. Makroo, and D. Dahiya, "Round Robin with server affinity: A VM load balancing algorithm for cloud based infrastructure," *J. Inf. Process. Syst.*, vol. 9, no. 3, pp. 379–394, Sep. 2013.

[7] P. Sangwan, M. Sharma, and A. Kumar, "Improved Round Robin scheduling in cloud computing," *Adv. Comput. Sci. Technol.*, vol. 10, no. 4, pp. 639–644, 2017.

[8] D. C. Devi and V. R. Uthariaraj, "Load balancing in cloud computing environment using improved weighted Round Robin algorithm for nonpreemptive dependent tasks," *Sci. World J.*, vol. 2016, Feb. 2016, Art. no. 3896065.

[9] E. Qin, Y. Wang, L. Yuan, and Y. Zhong, "Research on Nginx dynamic load balancing algorithm," in *Proc. 12th Int. Conf. Measuring Technol. Mechatronics Autom. (ICMTMA)*, Feb. 2020, pp. 620–624.

[10] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning," *Appl. Soft Comput.*, vol. 76, pp. 416–424, Mar. 2019.

[11] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," *Proc. Comput. Sci.*, vol. 115, pp. 322–329, 2017.

[12] M. Adhikari and T. Amgoth, "Heuristic-based load-balancing algorithm for IaaS cloud," *Future Gener. Comput. Syst.*, vol. 81, pp. 156–165, Apr. 2018.

[13] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment," *Proc. Comput. Sci.*, vol. 151, pp. 519–526, Jan. 2019.

[14] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146379–146389, 2019.

[15] A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 10, pp. 3975–3987, Oct. 2020.

[16] S. Velliangiri, P. Karthikeyan, V. M. Arul Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing," *Ain Shams Eng. J.*, vol. 12, no. 1, pp. 631–639, Mar. 2021.

[17] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic bio-inspired load balancing algorithm in cloud computing," *IEEE Access*, vol. 7, pp. 42735–42744, 2019.

[18] Y. Zhao, H. Liu, Y. Wang, Z. Zhang, and D. Zuo, "Reducing the upfront cost of private clouds with clairvoyant virtual machine placement," *J. Supercomput.*, vol. 75, no. 1, pp. 340–369, Jan. 2019.

[19] S. Xue, Y. Peng, X. Xu, J. Zhang, C. Shen, and F. Ruan, "DSM: A dynamic scheduling method for concurrent workflows in cloud environment," *Cluster Comput.*, vol. 22, no. S1, pp. 693–706, Jan. 2019.

[20] N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," *Comput. Commun.*, vol. 187, pp. 35–44, Apr. 2022.

[21] A. M. S. Kumar and M. Venkatesan, "Task scheduling in a cloud computing environment using HGPSO algorithm," *Cluster Comput.*, vol. 22, no. S1, pp. 2179–2185, Jan. 2019.

[22] G. A. P. Princess and A. S. Radhamani, "A hybrid meta-heuristic for optimal load balancing in cloud computing," *J. Grid Comput.*, vol. 19, no. 2, pp. 1–22, Jun. 2021.

[23] N. Manikandan, P. Divya, and S. Janani, "BWFSO: Hybrid black-widow and fish swarm optimization algorithm for resource allocation and task scheduling in cloud computing," *Mater. Today, Proc.*, vol. 62, pp. 4903–4908, Jan. 2022.

[24] F. N. Al-Wesabi, M. Obayya, M. A. Hamza, J. S. Alzahrani, D. Gupta, and S. Kumar, "Energy aware resource optimization using unified metaheuristic optimization algorithm allocation for cloud computing environment," *Sustain. Comput., Informat. Syst.*, vol. 35, Sep. 2022, Art. no. 100686.

[25] B. Kruekaew and W. Kimpan, "Multi-objective task scheduling optimization for load balancing in cloud computing environment using hybrid artificial bee colony algorithm with reinforcement learning," *IEEE Access*, vol. 10, pp. 17803–17818, 2022.

[26] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," *Cluster Comput.*, vol. 23, no. 4, pp. 2891–2899, Dec. 2020.

[27] A. Moori, B. Barekatain, and M. Akbari, "LATOC: An enhanced load balancing algorithm based on hybrid AHP-TOPSIS and OPSO algorithms in cloud computing," *J. Supercomput.*, vol. 78, no. 4, pp. 4882–4910, Mar. 2022.

[28] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2332–2342, Jun. 2020.

[29] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[30] A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: Multi-resource load balancing algorithm for cloud computing using ant colony optimization," *Cluster Comput.*, vol. 24, no. 4, pp. 3135–3145, Dec. 2021.

[31] P. Verma, S. Shrivastava, and R. K. Pateriya, "Enhancing load balancing in cloud computing by ant colony optimization method," *Int. J. Comput. Eng. In Res. Trends*, no. 4, pp. 277–284, Jun. 2017.

[32] X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 113–128, Feb. 2018.

[33] H. Zhao, J. Wang, F. Liu, Q. Wang, W. Zhang, and Q. Zheng, "Power-aware and performance-guaranteed virtual machine placement in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1385–1400, Jun. 2018.

[34] H. Xing, J. Zhu, R. Qu, P. Dai, S. Luo, and M. A. Iqbal, "An ACO for energy-efficient and traffic-aware virtual machine placement in cloud computing," *Swarm Evol. Comput.*, vol. 68, Feb. 2022, Art. no. 101012.

[35] O. Y. Abdulhammed, "Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm," *J. Supercomput.*, vol. 78, no. 3, pp. 3266–3287, 2022.

[36] J. Devagnanam and N. M. Elango, "Design and development of exponential lion algorithm for optimal allocation of cluster resources in cloud," *Cluster Comput.*, vol. 22, no. S1, pp. 1385–1400, Jan. 2019.

[37] V. M. Arul Xavier and S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. S1, pp. 287–297, Jan. 2019.

[38] A. Jangra and N. Mangla, "An efficient load balancing framework for deploying resource schedulingin cloud based communication in healthcare," *Meas., Sensors*, vol. 25, Feb. 2023, Art. no. 100584.

[39] 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, Mar. 2022.

[40] A. N. Aliyu and P. B. Souley, "Performance analysis of a hybrid approach to enhance load balancing in a heterogeneous cloud environment," *Int. J. Adv. Sci. Res. Eng.*, vol. 5, no. 7, pp. 246–257, 2019.

[41] D. A. Shafiq, N. Z. Jhanjhi, A. Abdullah, and M. A. Alzain, "A load balancing algorithm for the data centres to optimize cloud computing applications," *IEEE Access*, vol. 9, pp. 41731–41744, 2021.

[42] S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2370–2382, Jun. 2022.

[43] S. Negi, M. M. S. Rauthan, K. S. Vaisla, and N. Panwar, "CMODLB: An efficient load balancing approach in cloud computing environment," *J. Supercomput.*, vol. 77, no. 8, pp. 8787–8839, Aug. 2021.

[44] J. P. B. Mapetu, L. Kong, and Z. Chen, "A dynamic VM consolidation approach based on load balancing using Pearson correlation in cloud computing," *J. Supercomput.*, vol. 77, no. 6, pp. 5840–5881, Jun. 2021.

[45] A. Semmoud, M. Hakem, B. Benmammar, and J. Charr, "Load balancing in cloud computing environments based on adaptive starvation threshold," *Concurrency Comput.*, vol. 32, no. 11, p. e5652, Jun. 2020.

[46] N. S. Kaurav and P. Yadav, "A genetic algorithm based load balancing approach for resource optimization for cloud computing environment," *Int. J. Inf. Comput. Sci.*, vol. 6, no. 3, pp. 175–184, Mar. 2019.

[47] M. A. Alamin, M. K. Elbashir, and A. A. Osman, "A load balancing algorithm to enhance the response time in cloud computing," *Red Sea Univ. J. Basic Appl. Sci.*, vol. 2, no. 2, pp. 473–490, 2017.

[48] 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, pp. 64–77, Feb. 2019.

[49] S. P. M. Ziyath and S. Senthilkumar, "MHO: Meta heuristic optimization applied task scheduling with load balancing technique for cloud infrastructure services," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 6, pp. 6629–6638, Jun. 2021.

[50] A. Vinothini and P. Balasubramanie, "Meta-heuristic firefly approach to multi-servers load balancing with independent and dependent server availability consideration," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 5, pp. 5443–5455, 2021.

[51] I. Attiya, M. A. Elaziz, and S. Xiong, "Job scheduling in cloud computing using a modified Harris hawks optimization and simulated annealing algorithm," *Comput. Intell. Neurosci.*, vol. 2020, Mar. 2020, Art. no. 3504642.

[52] G. Li and Z. Wu, "Ant colony optimization task scheduling algorithm for SWIM based on load balancing," *Future Internet*, vol. 11, no. 4, p. 90, Apr. 2019.

[53] 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, Oct. 2019.

[54] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. Int. Conf. Neural Netw. (ICNN)*, vol. 4, pp. 1942–1948, Nov./Dec. 1995.

[55] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, "IPSO task scheduling algorithm for large scale data in cloud computing environment," *IEEE Access*, vol. 7, pp. 5412–5420, 2019.

[56] R. M. Alguliyev, Y. N. Imamverdiyev, and F. J. Abdullayeva, "PSO-based load balancing method in cloud computing," *Autom. Control Comput. Sci.*, vol. 53, no. 1, pp. 45–55, 2019.

[57] M. Kumar and S. C. Sharma, "PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12103–12126, Aug. 2020.

[58] M. Kumar, K. Dubey, and S. C. Sharma, "Elastic and flexible deadline constraint load balancing algorithm for cloud computing," *Proc. Comput. Sci.*, vol. 125, pp. 717–724, Jan. 2018.

[59] V. Borovskiy, J. Wust, C. Schwarz, W. Koch, and A. Zeier, "A linear programming approach for optimizing workload distribution in a cloud," in *Proc. Int. Conf. Cloud Comput.*, 2011, pp. 127–132.

[60] V. Polepally and K. S. Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing," *Cluster Comput.*, vol. 22, no. S1, pp. 1099–1111, Jan. 2019.

[61] S. S. Rajput and V. S. Kushwah, "A genetic based improved load balanced min–min task scheduling algorithm for load balancing in cloud computing," in *Proc. 8th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2016, pp. 668–677.

[62] L. Tang, Z. Li, P. Ren, J. Pan, Z. Lu, J. Su, and Z. Meng, "Online and offline based load balance algorithm in cloud computing," *Knowl.-Based Syst.*, vol. 138, pp. 91–104, Dec. 2017.

[63] Z. Xiao, Z. Tong, K. Li, and K. Li, "Learning non-cooperative game for load balancing under self-interested distributed environment," *Appl. Soft Comput.*, vol. 52, pp. 376–386, Mar. 2017.

[64] L. D. D. Babu and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, May 2013.

[65] F. Ramezani, J. Lu, and F. K. Hussain, "Task-based system load balancing in cloud computing using particle swarm optimization," *Int. J. Parallel Program.*, vol. 42, no. 5, pp. 739–754, Oct. 2014.

[66] M. Vanitha and P. Marikkannu, "Effective resource utilization in cloud environment through a dynamic well-organized load balancing algorithm for virtual machines," *Comput. Electr. Eng.*, vol. 57, pp. 199–208, Jan. 2017.

[67] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Proc. Technol.*, vol. 10, pp. 340–347, Jan. 2013.

[68] K.-M. Cho, P.-W. Tsai, C.-W. Tsai, and C.-S. Yang, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing," *Neural Comput. Appl.*, vol. 26, no. 6, pp. 1297–1309, Aug. 2015.

[69] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," in *Proc. 6th Annu. Chinagrid Conf.*, Aug. 2011, pp. 3–9.

[70] A. Singh, D. Juneja, and M. Malhotra, "Autonomous agent based load balancing algorithm in cloud computing," *Proc. Comput. Sci.*, vol. 45, pp. 832–841, Jan. 2015.

[71] S. K. Vasudevan, S. Anandaram, A. J. Menon, and A. Aravinth, "A novel improved honey bee based load balancing technique in cloud computing environment," *Asian J. Inf. Technol.*, vol. 15, no. 9, pp. 1425–1430, 2016.

[72] M. Lavanya and V. Vaithiyanathan, "Load prediction algorithm for dynamic resource allocation," *Indian J. Sci. Technol.*, vol. 8, no. 35, pp. 1–4, Dec. 2015.

[73] R. Kapur, "A workload balanced approach for resource scheduling in cloud computing," in *Proc. 8th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2015, pp. 36–41.

[74] A. D. E. Michel, N. B. E. Noelle, and T. O. N. Y. E. Emmanuel, "Artificial bee colonies solution for load sharing in a cloud RAN," *Eur. J. Appl. Sci.*, vol. 10, no. 2, pp. 33–50, 2022.

[75] P. Kumar and R. Kumar, "Issues and challenges of load balancing techniques in cloud computing: A survey," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–35, Nov. 2019.

[76] K. T. Rajgopal, K. R. A. Kumar, and N. Shenoy, "Load balancing in cloud computing: A survey on popular techniques and comparative analysis," *Global J. Comput. Sci. Technol.*, vol. 18, pp. 1–11, Jun. 2018.

[77] S. Joshi and U. Kumari, "Load balancing in cloud computing: Challenges & issues," in *Proc. 2nd Int. Conf. Contemp. Comput. Informat. (IC3I)*, Dec. 2016, pp. 120–125.

[78] X. Feng, J. Ma, S. Liu, Y. Miao, and X. Liu, "Auto-scalable and fault-tolerant load balancing mechanism for cloud computing based on the proof-of-work election," *Sci. China Inf. Sci.*, vol. 65, no. 1, Jan. 2022, Art. no. 112102.

[79] P. P. G. Gopinath and S. K. Vasudevan, "An in-depth analysis and study of load balancing techniques in the cloud computing environment," *Proc. Comput. Sci.*, vol. 50, pp. 427–432, Jan. 2015.

[80] K. Kaur and R. Mahajan, "Equally spread current execution load algorithm—A novel approach for improving data centre's performance in cloud computing," *Int. J. Future Revolution Comput. Sci. Commun. Eng.*, vol. 4, no. 8, p. 8, Aug. 2018.

[81] M. O. Ahmad and R. Z. Khan, "Load balancing tools and techniques in cloud computing: A systematic review," in *Advances in Computer and Computational Sciences*. Singapore: Springer, 2018, pp. 181–195.

[82] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Load-balancing algorithms in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 88, pp. 50–71, Jun. 2017.

[83] T. Tamilvizhi and B. Parvathavarthini, "A novel method for adaptive fault tolerance during load balancing in cloud computing," *Cluster Comput.*, vol. 22, no. S5, pp. 10425–10438, Sep. 2019.

[84] A. S. Milani and N. J. Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends," *J. Netw. Comput. Appl.*, vol. 71, pp. 86–98, Aug. 2016.

[85] P. Xu, G. He, Z. Li, and Z. Zhang, "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization," *Int. J. Distrib. Sensor Netw.*, vol. 14, no. 12, Dec. 2018, Art. no. 155014771879379.

[86] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: A state of the art survey," *Int. J. Mod. Educ. Comput. Sci.*, vol. 8, no. 3, pp. 64–78, Mar. 2016.

[87] N. Thapliyal and P. Dimri, "Load balancing in cloud computing based on honey bee foraging behavior and load balance min-min scheduling algorithm," *Int. J. Electr. Electron. Res.*, vol. 10, no. 1, pp. 1–6, Mar. 2022.

[88] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *KSII Trans. Internet Inf. Syst.*, vol. 11, no. 12, pp. 5694–5711, 2017.

[89] A. Mallikarjuna and P. V. Krishna, "A nature-inspired approach for load balancing of tasks in cloud computing using equal time allocation," *Int. J. Innov. Technol. Exploring Eng. (IJITEE)*, vol. 8, no. 2S2, pp. 46–50, 2018.

[90] B. Mallikarjuna and P. V. Krishna, "OLB: A nature inspired approach for load balancing in cloud computing," *Cybern. Inf. Technol.*, vol. 15, no. 4, pp. 138–148, Nov. 2015.

[91] P. Nayak, J. Vania, and R. Robin, "Load balancing using modified throttled algorithm," *Int. J. Sci. Res.*, vol. 3, no. 3, pp. 3614–3616, 2015.

[92] S. R. Gundu and T. Anuradha, "Improved hybrid algorithm approach based load balancing technique in cloud computing," *Global J. Comput. Sci. Technol.*, vol. 19, no. 2, pp. 1–9, 2019.

[93] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment," *Int. J. Comput. Appl.*, vol. 42, no. 1, pp. 108–117, Jan. 2020.

[94] G. Soni and M. Kalra, "A novel approach for load balancing in cloud data center," in *Proc. IEEE Int. Advance Comput. Conf. (IACC)*, Feb. 2014, pp. 807–812.

[95] A. Tomar, B. Pant, V. Tripathi, P. Pandey, and K. K. Verma, "Improved task scheduling using effective Particle Swarm optimization in cloud computing environment," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 2, pp. 1232–1237, 2019.

[96] S. Pattnaik, J. P Mishra, B. K Sahoo, and B. K. Pattanayak, "Load balancing in cloud computing environment using CloudSim," in *Smart Innovation, Systems and Technologies*. Singapore: Springer, 2021, pp. 197–205.

[97] C. Xu and C. Song, "Optimization of innovation and entrepreneurship education and training system in colleges and universities based on OpenStack cloud computing," *Sci. Program.*, vol. 2022, Aug. 2022, Art. no. 2868499.

[98] S. Afzal and G. Kavitha, "Load balancing in cloud computing—A hierarchical taxonomical classification," *J. Cloud Comput.*, vol. 8, no. 1, pp. 1–4, Dec. 2019.

[99] S. H. H. Madni, M. S. A. Latiff, J. Ali, and S. M. Abdulhamid, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," *Arabian J. Sci. Eng.*, vol. 44, no. 4, pp. 3585–3602, Apr. 2019.

[100] F. Faridi, H. Sarwar, M. Ahtisham, S. Kumar, and K. Jamal, "Cloud computing approaches in health care," *Mater. Today, Proc.*, vol. 51, pp. 1217–1223, Jan. 2022.

[101] H. Ahmadvand, T. Dargahi, F. Foroutan, P. Okorie, and F. Esposito, "Big data processing at the edge with data skew aware resource allocation," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Heraklion, Greece, Nov. 2021, pp. 81–86.

[102] H. Ahmadvand, F. Foroutan, and M. Fathy, "DV-DVFS: Merging data variety and DVFS technique to manage the energy consumption of big data processing," *J. Big Data*, vol. 8, no. 1, pp. 1–16, Dec. 2021.

[103] H. Ahmadvand and M. Goudarzi, "SAIR: Significance-aware approach to improve QoR of big data processing in case of budget constraint," *J. Supercomput.*, vol. 75, no. 9, pp. 5760–5781, Sep. 2019.

[104] F. Hossein, "DV-ARPA: Data variety aware resource provisioning for big data processing in accumulative applications," 2020, *arXiv:2008.04674*.

[105] H. Ahmadvand, M. Goudarzi, and F. Foroutan, "Gapprox: Using gallup approach for approximation in big data processing," *J. Big Data*, vol. 6, no. 1, pp. 1–24, Dec. 2019.

[106] H. Ahmadvand and M. Goudarzi, "Using data variety for efficient progressive big data processing in warehouse-scale computers," *IEEE Comput. Archit. Lett.*, vol. 16, no. 2, pp. 166–169, Jul. 2017.

**YOGESH LOHUMI** is currently pursuing the M.Tech. degree in computer science and engineering with Graphic Era (Deemed to be University), Dehradun. With a deep fascination budget constraint for cutting-edge technologies, he specializes in the fields of cloud computing, artificial intelligence, and computational astrophysics. Throughout his academic journey, he has exhibited a steadfast dedication to academic excellence and a fervent aspiration to contribute to the advancement of knowledge in his areas of expertise. He is determined to make noteworthy contributions to the realms of cloud computing and artificial intelligence. He is currently associated with Graphic Era (Deemed to be University) and also an Assistant Professor with the Department of Computer Science and Engineering.

**DURGAPRASAD GANGODKAR** (Member, IEEE) received the B.E. degree (Hons.) from Karnataka University, the M.Tech. degree (Hons.) from Visvesvaraya Technological University, and the Ph.D. degree in computer science and engineering from IIT Roorkee, India. He was a recipient of IITR-Microsoft Research Grant for excellent research work and a Fellowship from AICTE during the Ph.D. degree. He is currently associated with Graphic Era (Deemed to be University) and the Dean of International Affairs and a Professor with the Department of Computer Science and Engineering and is involved in developing academic and research collaborations with universities and institutions abroad. He has been actively involved in research and has guided research scholars in the areas related to cloud and high-performance computing, machine learning, big data storage and security, and computer vision. He has contributed many novel techniques in varied fields of research with publications in reputed international conferences and high impact factor journals, including IEEE Transactions, Springer, Elsevier, Taylor, and Francis. He has chaired sessions in international conferences and has been a PC member of international conferences in South Korea, Spain, Italy, China, Austria, and India. He is a member of IEEE-USA, ACM-USA, and Computer Society of India and a Life Member of Indian Society for Technical Education.

**PRAKASH SRIVASTAVA** (Member, IEEE) received the M.E. degree in computer science and engineering from NITTTR Chandigarh, India, and the Ph.D. degree in computer science and engineering from the MMM University of Technology, Gorakhpur (previously affiliated to AKTU Lko), in 2017, under TEQIP II. He is currently an Associate Professor with the Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun, India. Prior to this assignment, he was with the KIET Group of Institutions, Ghaziabad, Amity University Noida, and Invertis University, Bareilly. He has published a large number of various research papers in reputed international journals and conferences of high repute. His research interests include computer networks, mobile ad-hoc networks, QoS, and cloud computing and data.

**MOHAMMAD ZUBAIR KHAN** (Senior Member, IEEE) received the Master of Technology degree in computer science and engineering from U. P. Technical University, Lucknow, India, and the Ph.D. degree in computer science and information technology from the Faculty of Engineering, M. J. P. Rohilkhand University, Bareilly, India. He was the Head and an Associate Professor with the Department of Computer Science and Engineering, Invertis University, Bareilly. He has more than 18 years of teaching and research experience. He is currently a Professor with the Department of Computer Science, Taibah University, Medina, Saudi Arabia. He has published more than 90 journals and conference papers. His current research interests include data mining, big data, parallel and distributed computing, theory of computations, and computer networks. He has been a member of the Computer Society of India, since 2004.

**ABDULRAHMAN ALAHMADI** received the Ph.D. degree in computer science and engineering from Southern Illinois University at Carbondale, in 2019. He is currently an Assistant Professor with the Department of Computer Science, Taibah University, Saudi Arabia. During the Ph.D. degree, he was with the Cloud Computing and Big Data Research Laboratory for five years. His Ph.D. research was in cloud computing data centers scheduling for energy consumption reduction and resource utilization improvement. Since then, he has published various peer-reviewed research papers in edge and fog cloud computing. His research interests include in machine learning resource management in cloud computing, task scheduling in fog computing, and the IoT-supported edge offloading techniques.

**AHMED H. ALAHMADI** received the Ph.D. degree in computer science and engineering from La Trobe University. His Ph.D. research was in e-health business requirements engineering. Since then, he has published various peer-reviewed research articles. He was the Dean of the College of Computer Science and IT, Albaha University, Saudi Arabia. He is currently an Associate Professor with the Department of Computer Science, Taibah University, Medina, Saudi Arabia, where he is also the CEO of the Applied College. In addition to research, he is also skilled in accreditation and college recruiting. He has made significant contributions in various research areas, including e-health, software engineering, business process modeling, requirements engineering, and process mining. He also has a demonstrated history of working in the higher education industry.