Received 22 December 2023, accepted 6 January 2024, date of publication 10 January 2024,
date of current version 18 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3352078

RESEARCH ARTICLE

A Comparative Analysis of Metaheuristic Techniques for High Availability Systems

DARAKHSHAN SYED[1], GHULAM MUHAMMAD SHAIKH[1], HANI MOHAMMED ALSHAHRANI[2], MOHAMMED HAMDI[2], MOHAMMAD ALSULAMI[2], ASADULLAH SHAIKH[3], (Senior Member, IEEE), AND SYED RIZWAN[4]

[1]Computer Science Department, Bahria University, Karachi 75260, Pakistan
[2]Department of Computer Science, College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia
[3]Department of Information Systems, College of Computer Science and Information Systems, Najran University, Najran 61441, Saudi Arabia
[4]Department Computer Science, Iqra University, Karachi 75500, Pakistan

Corresponding author: Darakhshan Syed (darakshansyed.bukc@bahria.edu.pk)

This work was supported by the Deanship of Scientific Research at Najran University are funding this work under the Research Groups Funding program grant code (NU/RG/SERC/12/27).

ABSTRACT In the ever-evolving technological landscape, ensuring high system availability has become a paramount concern. This research paper focuses on cloud computing, a domain witnessing exponential growth and emerging as a critical use case for high-availability systems. To fulfil the criteria, many services in cloud infrastructures should be combined, relying on the user's demands. Central to this study is load balancing, an integral element in harnessing the full potential of heterogeneous computing systems. In cloud environments, dynamic management of load balancing is crucial. This study explores how virtual machines can effectively remap resources in response to fluctuating loads dynamically, optimizing overall network performance. The core of this research involves an in-depth analysis of several metaheuristic algorithms applied to load balancing in cloud computing. These include Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization, Artificial Bee Colony, and Grey Wolf Optimization. Utilizing CloudAnalyst, the study conducts a comparative analysis of these techniques, focusing on key performance metrics such as Total Response Time (TRT) and Data Center Processing Time (DCPT). The findings of this research offer insights into the varying behaviors of these algorithms under different cloud configurations and user retention levels. The ultimate aim is to pave the way for developing innovative load-balancing strategies in cloud computing. By providing a comprehensive evaluation of existing metaheuristic methods, this paper contributes to advancing high-availability systems, underscoring the importance of tailored solutions in the dynamic realm of cloud technology.

INDEX TERMS Cloud computing, cloud analyst, high availability, load balancing, metaheuristics, performance analysis, swarm intelligence.

I. INTRODUCTION

A broad spectrum of users widely uses cloud computing to solve large-scale computing issues. Cloud computing services include heterogeneous computing resources such as CPUs, speed, and memory [1]. Confidentiality, load balancing, benchmarking, dynamic resources, sustainability, information security, optimal resource planning, data exchange cost, and energy efficiency are all current challenges in cloud computing. This study targets one of the most recent additions in the cloud computing world: load balancing. The production of quality of service (QoS) from several servers, referred to as a server data centre, is one of the most widely utilized load balancing uses. Load-balancing systems are commonly found on popular websites, large chat networks, high-bandwidth file transfer protocol sites, and DNS servers [2]. In practice, users would anticipate a particular level of QoS. Cloud vendors store the data in many cloud platforms or data centres (DCs). When consumers request services from cloud service providers, the desired tasks are

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

distributed across multiple servers using virtual machines (VM). These distinct VMs are allocated to different user duties. VMs are aided by cloud services, and all receiving load is spread among the various VMs. On several occasions, assigning work to different VMs can end in a few VMs being overloaded or needing more utilization [3]. Different load-balancing strategies derived from various literature are used to achieve the aim.

The objectives of this study are mentioned below:

- To investigate the effectiveness of several metaheuristic algorithms (including Genetic Algorithm, Particle Swarm Optimization, Ant Colony Optimization, Bat Algorithm, and Grey Wolf Optimization), particularly in the context of load balancing in cloud computing.
- To analyze the behaviour of the selected metaheuristic algorithms in various cloud infrastructure configurations and user retention scenarios using CloudAnalyst.
- To formulate guidelines and suggestions for creating novel load-balancing algorithms that can be used in cloud computing based on the knowledge gathered from the study.

In that way, this study encourages amateur researchers to participate easily in creating more effective load-balancing strategies. It will allow relevant academics to do more research in this area.

A. MOTIVATION

There are multiple motivations for attempting to write a research paper on the performance evaluation of various metaheuristic algorithms for load balancing in a cloud-based architecture, and the primary motivations are as follows [4], [5], [6], and [7]:

- To address the complexity of cloud environments because they are inherently complicated and dynamic. An encouraging substitute is metaheuristic algorithms, renowned for identifying ideal solutions in challenging search areas.
- To facilitate researchers to improve Cloud performance and efficiency by investigating different metaheuristic algorithms and recognizing approaches that significantly increase these metrics.
- To investigate the applicability of various algorithms. A comparative analysis reveals whether an approach better suits cloud environment types or load balancing problems.
- To encourage innovation in cloud computing through the comparative analysis of metaheuristic approaches. The study points out the advantages and disadvantages of the existing strategies, directing further advancements.
- Practical approaches are becoming necessary to satisfy the growing demands for Cloud Computing as more companies migrate to the cloud. This evaluation motivates the need to provide better technological solutions to fulfil these increasing needs.
- To evacuate knowledge gaps in the literature by thoroughly comprehending the various metaheuristic algorithms' performance. It contributes to the corpus of academic knowledge and is a valuable resource for other scholars and industry professionals.

This study makes a significant contribution to the field of cloud computing and may result in cloud services that are more efficient, cost-effective, and convenient. A comparison of the current research with prior studies is shown in Table 1. As indicated in Table 1, filling in the gaps found in the comparative study of the body of existing literature is essential to advancing the subject and guaranteeing thorough research. Not filling in these gaps may result in several consequences mentioned below [6], [8], [9], [10]:

- If the research may not fully explain how the algorithms perform under different scenarios and cloud settings if it does not cover varied real-world scenarios. Ultimately, it restricts the findings' application and might lead to solutions that are not tailored for various scenarios in real life.
- If graphical analysis and performance constraints are not considered, then the research might not be as thorough or transparent.
- If the researchers do not consider the control parameters, then they may not be able to offer practitioners helpful information to improve these algorithms for real world applications.
- More informed research may result in adequate conclusions in practical applications, particularly in crucial systems that depend on cloud computing for efficiency and performance.

Therefore, if these gaps are not filled, the research may not be as thorough, comprehensive, or practically applicable. In addition to bolstering the study, addressing these topics adds much to the corpus of information regarding cloud computing and metaheuristic algorithms.

B. OUR CONTRIBUTION

In the realm of algorithms used in high-availability systems, there has been a rise in the utilization of metaheuristic techniques, specifically Swarm Intelligence (SI) algorithms [11]. These algorithms have proven to be solutions for optimal load balancing. This research focuses on analyzing the efficiency of SI algorithms in improving load balancing and resource provisioning techniques, which are crucial for high-performance computing environments.

The study primarily explores known approaches such as Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC). Each technique offers unique perspectives and mechanisms for effectively distributing workloads [19], [20].

The main contribution of this work is to evaluate SI-based metaheuristic load-balancing algorithms under various scenarios. The behaviour of these algorithms is observed under

TABLE 1. Comparison of the presented study with the previous studies on metaheuristic-inspired load-balancing techniques.

Reference	Comparative Analysis	Varied Scenarios	Cloud Configuration	Graphical Analysis	Performance Constraints	Control Parameters
[12]	✓	✗	✗	✗	✗	✓
[13]	✓	✗	✓	✗	✗	✓
[14]	✓	✗	✗	✗	✗	✗
[15]	✓	✗	✓	✓	✗	✗
[16]	✓	✗	✓	✗	✓	✓
[17]	✓	✗	✓	✓	✓	✗
[18]	✓	✗	✓	✓	✗	✗
Presented Study	✓	✓	✓	✓	✓	✓

varied scenarios by using CloudAnalyst tool. The study's key metrics are Total Response Time (TRT) and Data Center Processing Time (DCPT). These metrics serve as benchmarks for comparing performance.

Moreover, this research goes beyond analysis by comparing the effectiveness of these metaheuristic techniques. It provides an overview based on response time and data centre processing times. Such a comprehensive comparison is essential for identifying the algorithms in terms of operational effectiveness and resource management. The main goal of this study is to guide researchers and professionals in the field of cloud computing for achieving optimal load balancing. By assessing these methods, the research helps make informed choices about selecting and using load-balancing algorithms in cloud-based settings. This contribution will play a significant role in advancing availability systems. The sections of this research article are divided as follows: Section II consists of the need for load balancing in the cloud environment. Section III presents the literature review of various metaheuristic-based load-balancing techniques for cloud. The comparative study of the discussed SI strategies based on performance and several quality metrics is presented in Section IV. The future possibilities and conclusions of SI algorithms utilized in cloud applications for load balancing are discussed in Section V and Section VI.

II. NEED OF LOAD BALANCING IN CLOUD COMPUTING

Load balancing is a method for evenly allocating additional dynamic workload among various endpoints. It's also utilized to boost user satisfaction and resource management and ensure that no single node is overloaded, resulting in better system reliability. We can use the available resources via load-balancing strategies by minimizing resource use (See Figure 1) [21]. Load balancing also strives to provide flexibility and agility for applications that may contribute to growth in the future and demand extra support, as well as to prioritize operations that need frequent execution above others.

Additional goals of load balancing include minimizing energy consumption and carbon emissions, preventing bottlenecks, configuration management, and achieving QoS

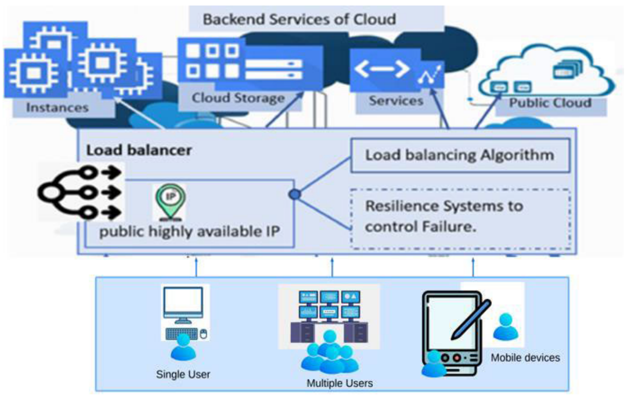

FIGURE 1. The architecture of load balancer in cloud computing.

standards for optimal task scheduling () [22], [23]. Workload mapping and load balancing methods that consider a variety of characteristics are necessary. A load balancer plays a vital role in enhancing cloud services' overall performance and reliability. Various QoS parameters are listed below:

- Resource Utilization: Load balancing helps evenly distribute tasks across all available servers, ensuring that every server is adequately utilized. It optimizes the use of hardware resources.
- Resilience: Load balancing makes adding new servers to the system easier, promoting scalability. To manage growing workloads without performance degradation, this is crucial.
- High Availability: It improves application availability by rerouting traffic from sick or overloaded servers to healthy ones, guaranteeing uninterrupted service availability.
- Cost-Effectiveness: Maximizing the usage of current resources through effective load balancing can save operating expenses by preventing the need for needless additional infrastructure.
- Optimal load Distribution: Load balancing ensures that tasks are divided according to each server's available memory, which results in high efficiency.

- Compliance: It can assist in meeting specific regulatory and compliance requirements by controlling where data is stored and processed.
- Improved User Experience: By ensuring applications are running smoothly and available when needed, load balancing enhances the end-user experience.

In summary, load balancing in cloud computing is not just about distributing workloads across multiple servers; it's a comprehensive strategy that touches upon various critical aspects of cloud service delivery, from operational efficiency and scalability to security and user satisfaction.

A. SERVICE MODELS OF CLOUD COMPUTING

A variety of service models are available to utilize cloud computing. These services are created to have specific qualities and meet the organization's needs. These models include Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [10].

- In the scope of SaaS, a service provider allows the use of one or more apps hosted on a cloud platform.
- In PaaS, a service provider provides users the resources to operate on cloud infrastructure and support for Software that the user has built or procured.
- The consumers in an IaaS model have been given the ability to control processes regulate storage, connection, and other fundamental computing resources. They are all helpful in managing arbitrary Software, including operating systems and apps.

B. TYPES OF CLOUD COMPUTING

One or more deployment strategies can be used to launch the services. To leverage various features of the cloud infrastructure, these models incorporate public, private, community, and hybrid cloud [24]. The public cloud infrastructure is made publicly available to major industrial groups or the public. In the case of the private cloud, the enterprise that designs the infrastructure cloud implementation is the only one with access to it. A third party or the business itself can handle private clouds. Numerous enterprises share a community cloud infrastructure and serve a specific community with everyday issues. It can be administered by the company or by a third party, and it can be implemented off-site or on-site. At the same time, there might be two or more clouds in a hybrid deployment paradigm, such as private, public, or community clouds.

C. CATEGORIZATION OF LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING

Two basic categories can be used to categorize load balancing in cloud services [25]:

1) STATIC LOAD BALANCING

The static approach needs to consider the base station's current position. All the connections, as well as their attributes, are known ahead of time. This sort of method's operation is based on predetermined knowledge. It is simple to implement and is independent of appropriate levels of real-time data [10].

2) DYNAMIC LOAD BALANCING

The dynamic balancing strategy considers the state of the system at run time. Changes in the structure of nodes are the basis for its operation. Dynamic strategies are challenging to construct, but they optimally distribute resources and optimize load once in place.

III. LITERATURE REVIEW

Many studies have been conducted to investigate and analyze SI-based task scheduling methods in a cloud-based framework for optimal load balancing [4]. Deepa and Cheelu [26] presented a comparative study on static and dynamic load-balancing algorithms. The static algorithms work correctly in an environment with a low load shift. Round robin (RR), Min-Min, Max-Min, and Opportunistic Load balancing algorithms are examples of static algorithms. Chaczko et al. [27] discussed RR as the most straightforward scheduling algorithm that promotes the idea of time frames. In this approach, the total available time is sliced into different slots. Each VM has a fixed slot to accomplish the assigned task(s). Ultimately, a user can utilize only the given time slot. If the task is not completed within the allocated period, the client is interrupted and must wait for the next slot. In that way, this algorithm picks the load randomly, so there is no check for underutilization. A weighted round-robin was created to address some of the problems with the round-robin algorithm [28].

Jobs are dispersed based on the weights supplied to each server. Processors with more processing power are given a higher value. As a result, the servers with the higher weights will get more work. Servers will receive evenly distributed traffic if all weights are equivalent. At the very first instance, the Min-Min algorithm monitors and calculates the nodes' minimum completion time. This minimum completion time is then used to assign the task to a particular VM. This action will go in a loop until all jobs are allocated to the VMs.

On the contrary, Max-min [26] employed the inverse method of min-min, in which the task with the shortest completion time is prioritized. In this approach, the estimated completion time of all jobs is calculated initially, and then a job that requires the most time to finish is chosen. Opportunistic is a static load balancing algorithm that does not consider the VMs' execution time or network load. It assigns jobs to the VMs at random. Because it does not estimate the VMs' execution time, task processing requires much time.

According to the discussion, more than static strategies are needed for heavy load transmission. As a result, to avoid the loopholes in the static approach, dynamic algorithms are considered. These algorithms to distribute the load have distinct policies such as transfer policy, selection policy, position policy, and knowledge-based policy. In a dynamic algorithm, advanced knowledge about the resource and job is optional because the resource is continuously monitored [29]. Larka et al. [30] suggested an approach for reducing response time, expense, and throughput parameter optimization. Thomas et al. [31] used a dynamic approach to minimize makespan. However, they still need to minimize makespan time for many tasks. Additional dynamic algorithms that tackle the work scheduling problem in a cloud context use heuristic approaches such as the max-min algorithm and metaheuristic approaches such as ACO, GA, PSO and ABC.

Dan et al. [32] utilized a genetic algorithm (GA) in combination with a local search technique for optimal load distribution. The core contribution of the proposed technique is to minimize the makespan. It also lowers the number of VMs who will miss their schedules. Furthermore, when evaluated and confirmed against other current strategies, this combination strategy enhanced the response time of VMs. This algorithm ensures that the client job's QoS requirements are met. PSO is another dynamic algorithm inspired by a swarm of flying birds. The swarm of flying birds searches for a suitable landing spot, and determining which one to use is difficult [33]. A variety of elements influence the takeoff. The availability of food, as well as the threat of predators, are among these considerations. Particle swarm optimization integrates workload with decreased response time for every assigned task, according to measurements of the productivity of numerous different algorithms. PSO has a lower cost of computing and is straightforward to comprehend and apply [34], [35]. It also works flexibly to deal with the tradeoff involving convergence and divergence and, in the end, provides the best answers to the complex problem of load balancing in Cloud Computing. Ant colonies were the inspiration for ACO [36]. With a group of VMs, look for better alternatives to a particular problem. A weighted graph is created to discover the best potential pathway. The load balancer gradually produces solutions by travelling along the graph. A speculative solution creation technique is a set of constraints linked to nodes and networks, the values of which are altered at runtime by the balancer.

ABC [37] is one of the most popular swarm intelligence-based algorithms that efficiently solve complex optimization problems; however, there seems to be some progress in exploitation, but it is good in terms of exploration. ABC is an exhaustive search algorithm that inspires investigation and is used for severe challenges. GWO [38] is a job allocation approach designed to balance load optimally. This algorithm is based on how wolves hunt. The load-balancing approach is broken down into four stages, just as wolves live in a group of four. Mahato et al. [39] utilizes GWO to achieve optimal load balancing in cloud computing. The target of this is to reduce the makespan by continuously updating the load on VMs and number of incoming tasks.

The summary of the various popular metaheuristic algorithms used for load balancing in Cloud Computing based on control parameters, performance constraints and problem applied are presented in Table 2. The overview of the quality-of-service parameters targeted by various metaheuristic algorithms is presented in Table 3.

A case study of Amazon AWS [40] presents a novel approach to load balancing based on many parameters by utilizing the concept of BAT algorithm. The recommended algorithm affects how the workload-sharing and task allocation between VMs in a data centre unfold effectively. There are two distinct layers for modelling. The first section is the pre-classification of the jobs in ascending order using the "Bat-algorithm" meta-heuristic. The second is assigning work to a certain number of virtual machines (VMs) with about equivalent performance. The suggested approach provides the optimal amount of VMs to complete the task. Depending on the task scales, it distributes jobs among VMs with the certainty of a balanced load distribution. It increases the total number of possible assignments of tasks to VMs in either parallel or series manner. Lastly, consider the local queue. Additionally, it can move tasks from one level to the preceding one to avoid overloaded and underutilized VMs.

IV. TARGETED ALGORITHMS FOR LOAD BALANCING IN CLOUD COMPUTING BASED ON METAHEURISTICS

This study targets Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Bat Algorithm (BAT) and Gray Wolf Optimizer (GWO). Every approach has unique characteristics and can be used for multiple aspects of comparative analysis related to load balancing in cloud computing (See Table 2). Comprehending these attributes is essential to realize that every algorithm may exhibit distinct performance in diverse conditions.

GA is famous for reducing the makespan to a considerable degree. For example, ACO is well known for its effectiveness in path-finding and scheduling tasks, essential to cloud computing load balancing [41], [42], [43]. GWO is also gaining attention because of its improved convergence() [44], [45]. PSO is best suited for providing a globally optimized solution [46]. BAT is a well-suited algorithm for optimal resource utilization [47].

The rationale for choosing these algorithms is based on their inherent properties that align closely with the requirements of cloud load balancing. These algorithms were selected because of their intrinsic qualities, which closely match cloud load balancing requirements.

A. GENETIC ALGORITHM

According to Makasarwala et al. [48], the GA plays a critical role in load balancing. Prioritization is used to begin population activation. The time is used to determine the request's priority. The job's duration determines it. The longer the job, the more time it takes. The demand necessitates less time and is made earlier. Chromosomes are selected based on fitness functions. Based on this, mapping and swapping is performed. After that, a chromosome is added to the new population. It all will go in a loop until or unless the termination condition is met. This technique provides a better average response time. Zhou et al. [49] presented an improved version of GA by utilizing the concept of greedy strategy. The idea is to track all the VMs in a loop. In that way,

TABLE 2. Comparisons between the targeted algorithms based on performance constraints, control parameters and problems applied.

Criteria	GA	PSO	ACO	Bat	GWO
Based on performance constraints	Necessitates less knowledge of the targeted problem. Designing operators correctly can be difficult. Time consuming.	Slow convergence. Can stuck in local optimum.	Stagnation phase. Exploration and exploitation rate. Convergence speed is slow.	Multiple iterations are required to achieve the optimum solution.	Improved convergence speed. Limited solving accuracy. Poor local exploring potential.
Based on control parameters	Statistics of mutation and crossover. The fitness value scaling. The population's size.	The scope of the problem. Coefficients of acceleration. Number of particles. Neighbour-hood size. Number of iterations. The variables that scale the involvement of the cognitive and social elements.	Alpha Beta Evaporation rate	Population size No. of Maximum cycles .	Initial position. inertia weight. Adjustment coefficient. Inertial weight and other parameters.
Based on problems applied	Timetabling and scheduling problems. Global optimization problems. Problem domains that have a complex fitness landscape.	Optimization problems visual effects.	Solve challenging optimal solution situations using approximations.	High-dimensional problems. Optimization problems.	Optimize engineering design problems. NP-hard problems. Integer programming and minimax problems, Scheduling problems

if any VM is available at run time, the job can be shifted to it from any overloaded VM. All these calculations are made based on completion time. This method not only distributes the traffic in the most efficient way possible, but it also increases resource utilization. Basu et al. [50] proposed another improved technique of GA. Every chromosome in the population is regarded as a node in this approach. A node is assigned to each VM. Every node's VMs correspond to a chromosome's genetics. After performing crossover and mutation processes, optimization techniques were applied to acquire the resulting job allocation.

Saadat and Masehian [51] presented a hybridized bi-modular technique for load balancing in the cloud using a genetic algorithm. The proposed methodology has proven superior to previous techniques regarding load balancing and resource utilization. Although the second component emphasizes incorporating fuzziness, the first module applies Genetic Algorithm to identify the optimal task arrangements. The objective function was successfully accomplished based on their respective work queuing of detecting occupied server configurations. The suggested architecture offers a fuzzy performance for the availability of services.

GA provides robustness, reliability, and scalability in a cloud computing environment. Computer tests are also conducted in this study, with the best solution being displayed. As a result, the user acceptance rate improves. Gulbaaz et al. [52] introduced a revolutionary load-balancing system inspired by GA. This load balancer considers the actual load measured in a million instructions allocated to virtual machines (VMs).

Additionally stressed is the need for multi-objective optimization to optimize makespan and load balancing. Experiments use skewed, regular and uniform workload distributions with varying batch sizes. It has shown notable progress compared to this enhanced version of GA to other cutting-edge methods for makespan, throughput, and load balancing. Goar et al. [53] further advanced the Improved GA [52]. This Enhanced Improved GA is targeted to minimize the number of migrations. The proposed technique makes the shifting of load faster and more reliable by adjusting the estimation points for mutation. After performing tests, this algorithm is better in many QoS requirements than IGA [52].

B. PARTICLE SWARM OPTIMIZATION ALGORITHM

According to PSO, load balancing in Cloud systems is presented by Alguliyev et al. [54]. The approach for migrating tasks requires a lot of computational power to a high-performance virtual machine. Each scheduling

TABLE 3. Review of metaheuristic algorithms based on performance constraints.

Authors [Reference]	Algorithms	RT	T	MS	EC	S	RU
Makasarwala and Hazari [48]	GA	✗	✓	✓	✗	✗	✗
Gulbaaz et al. [52]	Improved GA	✗	✓	✓	✗	✗	✓
Alguliyev, et al. [54]	PSO	✓	✗	✗	✗	✗	✓
Golchi et al. [57]	IPSO	✓	✗	✓	✓	✗	✓
Valarmathi and Sheela [66]	Hybrid PSO	✓	✗	✓	✗	✓	✓
Li and Wu [60]	ACO	✓	✗	✗	✓	✗	✗
Ragmani et al. [62]	Improved Fuzzy ACO	✗	✗	✗	✓	✗	✓
Mosa et al. [67]	BAT	✗	✗	✓	✗	✗	✗
Muthsamy and Chandran [68]	Improved Bat	✓	✗	✓	✗	✗	✓
Gohil and Patel [69]	GWO	✓	✗	✓	✗	✗	✗
Sefati et al. [13]	Improved GWO	✓	✓	✓	✗	✓	✓

procedure is given a proportional weight to present its relevance. These weights eventually aid in identifying the best solution [55].

PSO is more effective when dealing with discrete problems than other optimization approaches [56]. Golchi et al. [57] targeted the issue associated with PSO, as PSO is particularly receptive to initiating a state that disturbs the results' quality. Therefore, an improved version of PSO is proposed. This hybrid approach uses the firefly algorithm to limit the range of search then in the second phase, optimal response is accomplished by using the Improved PSO.

Mapetu et al. [58] presented another variation in PSO as Binary PSO. The problem statement for this algorithm is to save time. The allocation of jobs is performed after analyzing the operational limitations. Its objective is to organize the VMs that users use to complete a task. With adequate load balancing and work scheduling, this notion gives scalability. Miao et al. [59] presented an Adaptive Pbest discrete PSO to overcome the randomness found in PSO. This algorithm first found all the reasonable solutions, saved them, and used them to update the best position of particles dynamically.

C. ANT_COLONY OPTIMIZATION ALGORITHM

ACO is a community-based metaheuristic technique for quickly evaluating viable alternatives to complicated problems [36]. Li and Wu [60] consider the quality requirement of users while assigning jobs. To improve the QoS the idea is to impose the Project management approaches with Simple ACO. The essential premise for choosing the appropriate job scheduling priority for the ant colony algorithm is used to achieve load balancing. In this context, the scheduler can be compared to an ant and the scheduling phase to an ant foraging activity.

Gupta and Garg [61] also promoted ACO for load balancing. The idea is to use the starting time of a particular task and compute the duration to complete that task. Using the completion of these tasks, the beginning time of the following task(s) can be determined. When all the activities are completed, the total completion time is used to assign tasks to VM(s).

The ant concept is used to assign tasks to only the available resources. The fitness function is assessed for each node to announce the search strategy. Then, after achieving the local search for every node, it eventually leads to a global search with better results.

Ragmani et al. [62] proposed Improved ACO by including fuzziness in ACO. This technique presents an optimal solution and is convenient for complex networks with massive load. Simulation results show that this approach reduces the response time to almost 81%. Junaid et al. [30] proposed a Hybrid ACO to achieve multi-task scheduling. This algorithm improves the QoS requirements by minimizing the makespan and migration time.

D. BAT ALGORITHM

BAT is an optimization technique based on the echolocation phenomenon. BATs utilize their hearing to track down their prey. When bats make sound, they travel to various prey available in frequency form. After collecting all of the signals, they determine the distance using this frequency [63]. The same concept may be applied to load balancing, where each node runs independently and focuses on keeping coordination localized. The BAT approach was tested in VM by Ullah et al. [47] to see if it enhances load balancing in cloud environments. The concept is that whenever a job requires entertainment, the best VM from all participating Nodes is considered initially.

An improved BAT algorithm is presented to produce additional optimum and finer outcomes [64]. The algorithm must be run iteratively for this to be feasible. The BAT algorithm identifies the best server among the accessible servers whenever a task needs to be processed. Simultaneously, the load scheduler determines the job form and resource requirements, as well as the best virtual machine (VM) for the execution of tasks. If the available server can meet the criteria, the load is allocated; if it is too high, it is divided among other servers. This method ensures load balancing by all servers occupied, ensuring they are neither underloaded nor overloaded. Not only does the proposed solution reduce response time, but it also executes load balancing with minimal delays.

E. GRAY WOLF OPTIMIZATION ALGORITHM

The GWO approach begins by gathering all task and resource details and determining whether or not the distribution requirements have been met [65]. The scheduler is allotted a resource after examining the best option. It aimed to minimize the time it takes to manufacture anything, the total cost, and the number of tasks that can be completed in the time allotted.

Gohil and Patel also suggest a combined approach of PSO and GWO [69] to achieve the best availability of VM(s) to balance load more efficiently. It is recommended in this strategy first to apply GWO to generate the best position as alpha, i.e. X_a. After that, rather than determining the perfect location, PSO is carried out with the help of alpha. As a result, establishing an objective function that efficiently manages load in a cloud environment is the technique's primary targeted problem. The purpose of the proposed technique is to add randomness in GWO to avoid premature convergence.

Xingjun et al. [70] presented a Fuzzy logic-based GWO Algorithm to make the technique more effective. The main target of this approach is to improve the response time by managing load. The algorithms first find the overloaded nodes. In case of the arrival of a request, the load is assigned to those VMs that are not overloaded recently. To make GWO stable, the researchers add fuzziness to the technique for better load adjustment. Despite achieving a better response, the convergence and degree of imbalance were not targeted as QoS parameters. Ouhame et al. [71] targeted the problem statement of failure in allocation method. In this first instance, this research highlights the main two categories i.e., overutilized and underutilized. The Gray wolf improves the search strategy initially before being used by the ABC algorithm to improve it further. As a whole, efficiency, stability, communication cost, and power consumption are all improved with this hybrid methodology.

Table 3 presents a review of specific metaheuristic algorithms based on performance constraints. Where; 'R' is the Response Time, 'T' is the Throughput, 'MS' is the Makespan, 'EC' is the Energy Conservation, 'S' is the Scalability, and 'RU' is the Resource Utilization. It can be summarized as:

- The algorithms that adhere to real-time constraints are PSO, Bat, and ACO. It implies that they might be appropriate for applications that need real-time processing or swift responses.
- High throughput is only demonstrated by GA and GWO. Specific algorithms may be more effective when processing significantly high traffic quickly.
- GA, PSO, BAT, and GWO possess improved makespan efficiency. It suggests that computers with fewer memory resources would benefit more from these techniques.
- The only algorithms that adhere to minimal energy consumption limits are ACO and GWO. In applications where energy is a concern, they might be selected.
- Both PSO and Improved GWO exhibit scalability, allowing them to grow to do additional tasks to VM assignments or to manage sudden traffic spikes.
- High resource utilization is observed for Improved GA, PSO, IPSO, Hybrid PSO, Fuzzy ACO, Improved Bat, and Improved GWO. It demonstrates that they can use system resources effectively.

From here, it can be concluded that different algorithms, such as PSO and GWO, exhibit distinct limitations and are scalable. At the same time, improved GA is noteworthy for its throughput and utilization of resources, but it's not scalable or energy efficient. PSO and its variants (PSO, IPSO, and Hybrid PSO) have good real-time performance and efficient use of resources but poor throughput and energy efficiency. The real-time performance, makespan efficiency, and resource utilization of GWO and its enhanced version are exceptional. Its modified version offers substantial throughput as well. The BAT algorithm and its enhanced version (although they lack other QoS) perform well in real-time and makespan.

V. RESEARCH METHODOLOGY

The research methodology is designed to provide a comprehensive and systematic evaluation of metaheuristic algorithms in the context of cloud computing, offering significant insights into their application for enhancing the performance and reliability of cloud-based systems (See Figure 2).

The study first selects several prominent metaheuristic algorithms for analysis, including the GA, PSO, ACO, Bat Algorithm, and GWO. The rationale for this choice is its applicability and possible effectiveness concerning load balancing in cloud computing configurations. The study simulates a cloud computing infrastructure by establishing a virtual environment with CloudAnalyst, a well-known simulation tool. With the help of this tool, a variety of configurations and scenarios typical of cloud systems in the actual world can be modelled. The CloudAnalyst environment is used to implement each of the chosen algorithms. It entails setting up the algorithms to specifically deal with the issue of load balancing amongst virtual machines in a cloud configuration. Every algorithm can dynamically remap resources in response to varying loads according to the implementation. The study evaluates each algorithm's performance through a series of experiments carried out with CloudAnalyst.

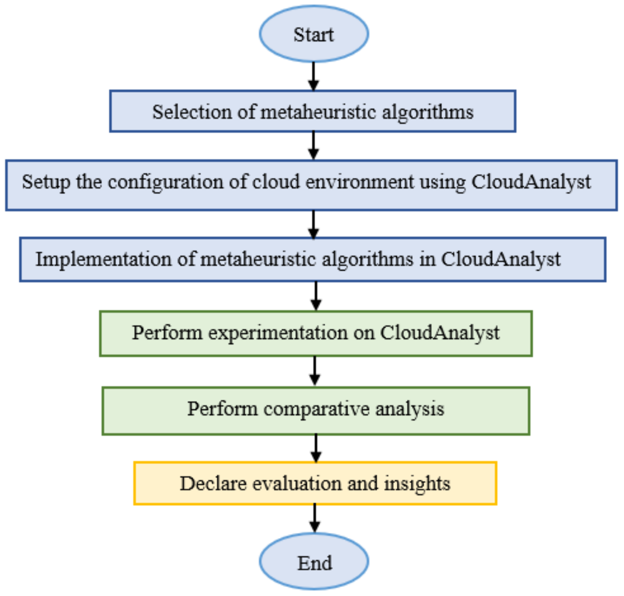
Start
Selection of metaheuristic algorithms
Setup the configuration of cloud environment using CloudAnalyst
Implementation of metaheuristic algorithms in CloudAnalyst
Perform experimentation on CloudAnalyst
Perform comparative analysis
Declare evaluation and insights
End

FIGURE 2. Process flow of the proposed methodology.

These tests simulate various load scenarios and cloud configurations to comprehend the behaviour of the algorithms under different circumstances. Significant performance parameters are recorded for each experiment, including Total Response Time (TRT) and Data Centre Processing Time (DCPT). The collected data is examined to evaluate how well the algorithms perform regarding various metrics. Finding each algorithm's advantages and disadvantages concerning response time and processing time across various cloud configurations is the primary goal of the analysis.

Based on the comparative analysis, the paper assesses how well each method performs in cloud computing environments in terms of establishing effective load balancing. We derive some insights into how these algorithms may be coupled or utilized to create novel load-balancing schemes that meet the changing needs of cloud computing environments. The study aims to provide practical insights that can guide the creation of more sophisticated and specialized load-balancing systems for cloud computing.

VI. PERFORMANCE EVALUATION AND CONSIDERATIONS

This section presents the performance analysis of the targeted metaheuristic-based load balancing algorithms in terms of overall response time and data centre processing time through simulations.

A. CLOUD ANALYST

For the experimental analysis, cloud analyst [72] is used, designed to help researchers and practitioners manage and simulate the cloud environment. It allows users to quickly and easily assess the requirements of large-scale Cloud applications in terms of geographic distribution [73]. It is made up of three parts: the data center (DC), the user base (UB), and the Internet (IR). The domains' representation of cloud analyst is demonstrated in Figure 3, while Figure 4 depicts the basic operation of load balancer on which SI-based algorithms are applied. Figure 5 shows the cloud analyst's environment picture [74]. It allows you to simulate DC deployment, UB availability, and how tasks are distributed to different DCs with different costs. This application aids in the management of appropriate load balancing across multiple UBs with accompanying DC and internet connections.

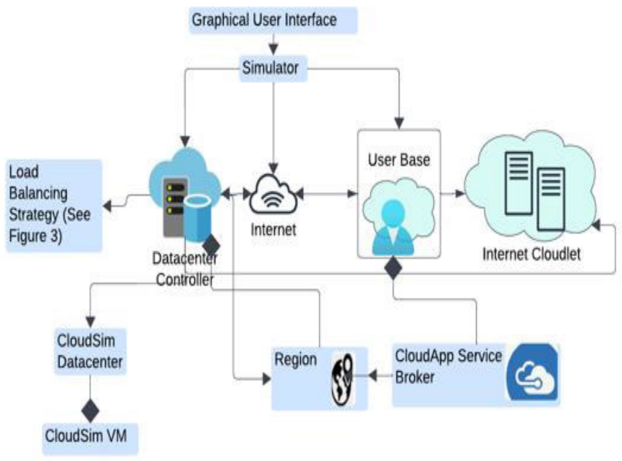

FIGURE 3. Domains' representation of CloudAnalyst.

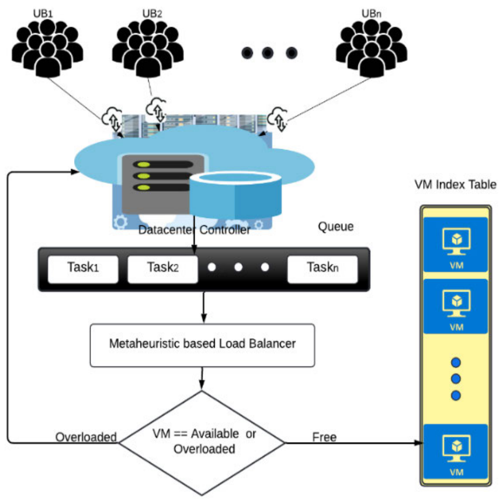

FIGURE 4. Working of load balancing algorithm in CloudAnalyst.

B. CONFIGURATION OF SIMULATION

The configuration of the various components of the cloud analyst tool is required to analyze various load-balancing policies. As shown in Tables 4-7, the settings for user base configuration, application deployment configuration, data

TABLE 4. Configuration of user bases.

Name	REGION	RPUPH	DSPR (bytes)	PHS (GMT)	PHE (GMT)	APU	AOPU
UB1	0	60	100	3	9	1000	100
UB2	1	60	100	3	9	1000	100
UB3	2	60	100	3	9	1000	100
UB4	3	60	100	3	9	1000	100
UB5	4	60	100	3	9	1000	100
UB6	5	60	100	3	9	1000	100

TABLE 5. Configuration of application deployment.

Data Center	NO. OF VMS	Image Size	Memory	Bandwidth
DC1	5	10000	512	1000
DC2	5	10000	512	1000
DC3	5	10000	512	1000
DC4	5	10000	512	1000
DC5	5	10000	512	1000
DC6	5	10000	512	1000

TABLE 6. Configuration of data centers.

Name	R	Architecture	OS	VM	CPV ($/Hr)	MC ($/s)	SC ($/s)	DC ($/Gb)	PHUs
DC1	0	X86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC2	1	X86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC3	2	X86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC4	3	X86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC5	4	X86	Linux	Xen	0.1	0.05	0.1	0.1	1
DC6	5	X86	Linux	Xen	0.1	0.05	0.1	0.1	1

TABLE 7. Physical hardware details.

ID	Memory (MB)	Storage (Mb)	Available	No. of Processors	Processor Speed	VM Policy
0	204800	100000000	1000000	4	10000	Time shared
1	204800	100000000	1000000	4	10000	Time shared
2	204800	100000000	1000000	4	10000	Time shared
3	204800	100000000	1000000	4	10000	Time shared

centre configuration, and load balancing policy have all been configured. For the various user bases, we chose different parts of the world. As in Figure 5, the simulated scenario shows six different datacenters and six user bases placed in different geographical locations on the map. Also, for the simulation, we chose 10 for the user grouping factor in user bases, 10 for request grouping factor in Data Centers and 100 for the executable instruction length per request.

Where; 'RPUPH' is the request per user per Hour, 'DSPR' is the Data size per request, 'PHS' is the Peak Hours Start, 'PHE' is the Peak Hours End, 'APU' is the Average Peak Users and 'AOPU' is the Average Off-Peak Users (See Table 4).

Where; 'OS' is the Operating System, 'VM' is the Virtual machine used, 'CPV' is the Cost per VM used, 'MC' is the Memory Cost, 'SC' is the Storage Cost, 'DTC' is the

TABLE 8. Comparison of load balancing algorithms in term of Overall Response Time (RT) (UB=10 and DC=5).

Algorithms	NO. OF UBS	No. of DCs	No. of VMs	Overall RT
Ant Colony Optimization	10	5	25	**100.02**
Particle Swarm Optimization	10	5	25	100.33
Genetic	10	5	25	100.65
Bat	10	5	25	100.50
Gray Wolf Optimization	10	5	25	100.21

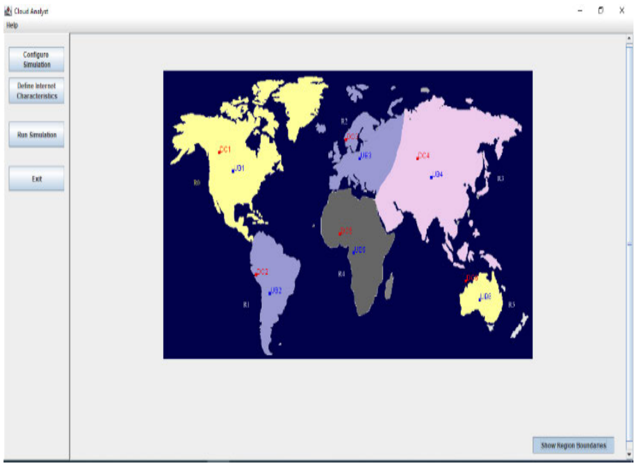

FIGURE 5. Various regions, user bases and data centres worldwide are depicted.

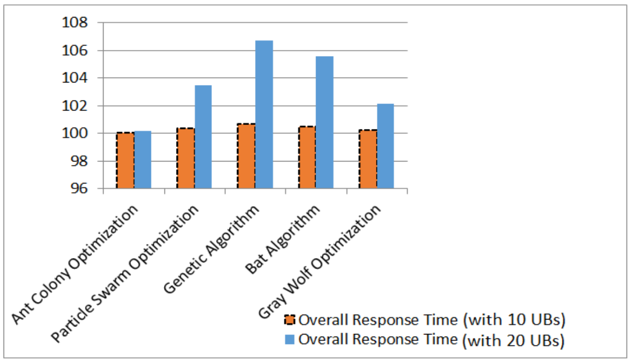

FIGURE 6. Overall Response Time when UBs=10 and 20 with DCs=5 and VMs = 25. (See Table 8 and 9).

TABLE 9. Comparison of load balancing algorithms in term of Overall Response Time (UB=20 and DC=5).

Algorithms	NO. OF UBS	No. of DCs	No. of VMs	Overall RT
Ant Colony Optimization	20	5	25	**100.15**
Particle Swarm Optimization	20	5	25	103.49
Genetic	20	5	25	106.68
Bat	20	5	25	105.59
Gray Wolf Optimization	20	5	25	102.14

Data Transfer Cost and 'PHU' is the Physical Hardware Unit (Table 6).

C. SIMULATION RESULTS

The best way to test an algorithm in cloud computing is through simulation and virtual experimentation. For each VM load balancing scheduling algorithm, the parameters described in the configuration section were employed individually, and the metrics were calculated based on the results. The testing criteria are divided into five scenarios to obtain the results.

1) SCENARIO 1

In the first scenario we have initialized user bases (UBs), datacenters (DCs) and virtual machines (VMs) as 10, 5 and 25, respectively (see Table 8). After analyzing it, we have found that based on response time, the performance of ACO outperforms all of them, as shown in Figure 6.

2) SCENARIO 2

In the second scenario, we have initialized the values user bases, data centers, and virtual machines as 20, 5 and 25, respectively (see Table 9). After analyzing it we have found that based on response time, the performance of ACO outperforms among all of them as shown in Figure 6.

Table 8 demonstrates that when the UB is 10, ACO is 0.31 % superior to PSO, 0.63 % better than GA, 0.48% better than Bat, and 0.19 % faster than GWO regarding overall response time. The acquired results of ACO over others enhance when the UB is increased from 10 to 20.

Table 9 demonstrates that when the UB is 20, ACO is 3.34% better than PSO, 6.52 % better than GA, 5.43 % better than Bat, and 1.99 % better than GWO in terms of overall response time.

3) SCENARIO 3

In the third scenario, we have initialized the values of user bases, data centres and virtual machines as 10, 5 and 25, respectively (see Table 10). After analyzing it we have found

TABLE 10. Comparison of load balancing algorithms in terms of overall data center processing time(dcpt) (UB=10 and DC=5).

Algorithms	No. OF UBs	No. of DCs	No. of VMs	Overall DCPT
Ant Colony Optimization	10	5	25	**5.56**
Particle Swarm Optimization	10	5	25	7.12
Genetic	10	5	25	8.61
Bat	10	5	25	9.39
Gray Wolf Optimization	10	5	25	6.06

TABLE 11. Comparison of load balancing algorithms in terms of overall data center processing time (DCPT) (UB=10 and DC=10).

Algorithms	No. OF UBs	No. of DCs	No. of VMs	Overall DCPT
Ant Colony Optimization	10	10	25	**8.3**
Particle Swarm Optimization	10	10	25	10.7
Genetic	10	10	25	12.7
Bat	10	10	25	13.9
Gray Wolf Optimization	10	10	25	9

that on the basis of datacenter processing time, the performance of ACO outperforms among all of them as shown in Figure 7.

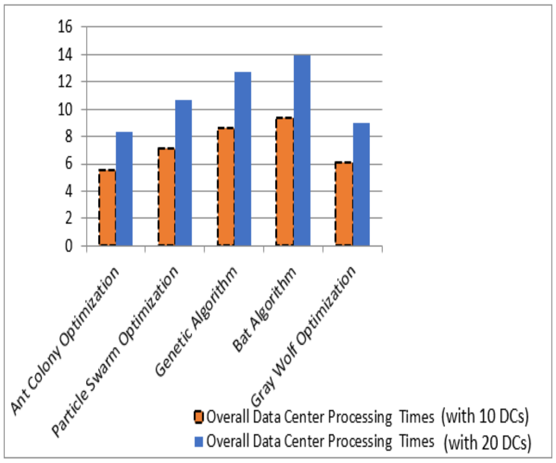

FIGURE 7. Overall data centre processing times when DCs=5 and 10 with UBs=5 and VMs = 25. (See Table 10 and 11).

4) SCENARIO 4

In the fourth scenario, we have initialized the values of user bases, data centres and virtual machines as 10, 10 and 25, respectively (see Table 11). After analyzing it, we have found that based on data centre processing time, the performance of ACO outperforms all of them, as shown in Figure 7. The algorithm successfully balances the load based on the accessibility of the central memory unit.

Table 10 demonstrates that when the DC is 10, ACO is 28.06 % superior to PSO, 54.85% better than GA, 68.88% better than Bat, and 8.99 % faster than GWO regarding overall data centre processing time. The acquired results of ACO over others enhance when the DC is increased from 10 to 20. Table 11 demonstrates that when the UB is 20, ACO is 28.92 % better than PSO, 53.01% better than GA, 67.47 % better than Bat, and 8.43% better than GWO in overall data centre processing time.

5) SCENARIO 5

The fifth scenario behaves as an exceptional scenario. Additionally, this scenario also highlights the main contribution of this work. We have varied the values of user bases from 20 to 200 and then to 500 and initialized the data centres and virtual machines as 5 and 10, respectively (see Table 12 and Table 13). After analyzing it, we have found that based on overall response time, the performance of ACO degrades in case of a spike in the incoming user requests, as shown in Figure 8, due to the issue of slow convergence. In case of a sudden spike in traffic, GWO outperforms the others. PSO and Bat Algorithms also show improved results. PSO proved its scalability by outperforming Bat, increasing user retention from 200 to 500.

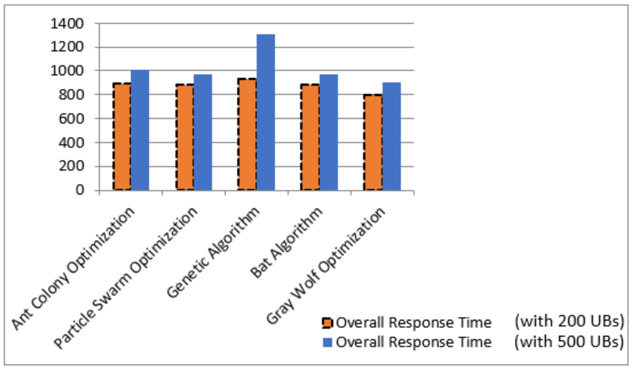

FIGURE 8. Overall response times when DCs=5 and VMs= 10 with UBs= 200 and 500, respectively. (See Table 12 and 13).

Table 12 demonstrates that when the incoming user requests are increased from 10 to 200, ACO degrades compared to the previous results. In this scenario, GWO outperforms. GWO is 11.88% superior to ACO, 10.86% better than PSO, 16.01% better than GA and 10.47% faster than BAT regarding overall response time.

The acquired results of GWO over others enhance when the traffic spike is further increased from 200 to 500. Table 13 demonstrates that when the UB is 500, GWO is 11.6 % better than ACO, GWO is 8.1 % better than PSO, 45.11% better than

TABLE 12. Comparison of load balancing algorithms in terms of overall response time (RT) (UB=500, DC=5 and VMs = 10).

Algorith ms	NO . OF UB S	No. of DC s	No. of V Ms	Overa ll RT
Ant Colony Optimization	200	5	25	895.1 6
Particle Swarm Optimization	200	5	25	887.2 5
Genetic	200	5	25	928.4 6
Bat	200	5	25	884.1 6
Gray Wolf Optimization	200	5	25	800.3 3

TABLE 13. Comparison of load balancing algorithms in terms of overall response time (RT) (UB=200, DC=5 and VMs = 10).

Algorithm s	No. of UB s	No. of DC s	No. of VM s	Overall RT
Ant Colony Optimization	500	5	25	1005.6 7
Particle Swarm Optimization	500	5	25	974.25
Genetic	500	5	25	1307.6 7
Bat	500	5	25	974.45
Gray Wolf Optimization	100	5	25	901.14

GA and 8.13% better than Bat in terms of overall response time.

6) COMPARATIVE PERFORMANCE ANALYSIS

From the results in Tables 8, 9, 10 and 11, we can interpret that no matter how many user bases and data centres are increased, ACO performs much superior among GWO, BAT, PSO and Genetic Algorithms. The performance of PSO and GA rises in tandem with the variety of tasks, but PSO is significantly less expensive than GA. Regarding overall response time and data processing time, GWO provides a higher level of service than PSO and GA.

On the other hand, Tables 12 and 13 (See Figure 8) conclude that in case of high user retention (i.e., when the UBs are increased to a considerable degree) ACO eventually degrades because of its slow convergence. In case of a sudden spike in traffic, GWO outperforms because of reduced convergence. Additionally, BAT provides improved optimal results in case of high user retention as compared PSO. But, in the longer term, when the user requests are further increased from 200 to 500 UBs, PSO outperforms BAT algorithm by providing a globally optimized solution. BAT degrades because of its premature convergence.

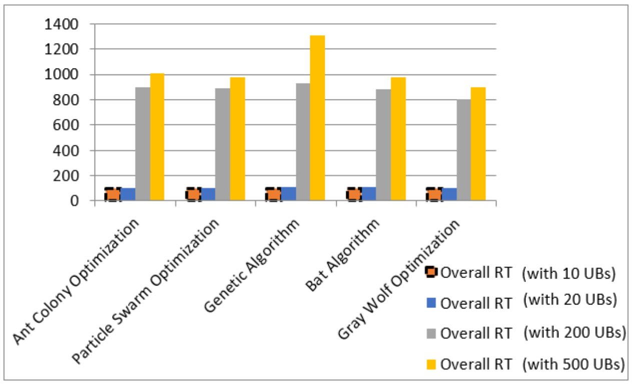

FIGURE 9. Variations in overall response times with respect to changing no. of UBs.

Figure 9 presents the variations in response time of the targeted algorithms with respect to the changing number of UBs. From this figure it is evident that when UBs rise from 10 to 20 in the case of ACO, the response time is largely constant. On the other hand, when UBs increase from 20 to 200 and 500, response times significantly increase. This implies that the ACO might find it difficult to effectively manage a high volume of incoming requests. In the case of PSO, response times increase in proportion to UBs, much like ACO does. The slow increase shows that PSO can grow at a faster rate than ACO. The response time increases noticeably as UBs are higher, indicating potential scalability issues. GA demonstrates a notable increase in response time when UBs increase from 20 to 200 and from 200 to 500. GA might therefore not handle a high rate of incoming requests efficiently. In contrast, the BAT Algorithm exhibits a consistent rise in response time and an increase in UBs, analogous to the PSO. At higher UB levels, there is a discernible increase in response time, although it appears to manage moderate increases in UBs very well. While GWO shows comparatively stable response times when the number of UBs rises from 10 to 20. As the incoming traffic rises from 20 to 200, the response time also increases, although not as significantly as in some other algorithms. When UBs hit 500, there's an obvious rise, but it's still lower than some other approaches.

VII. CONCLUSION AND FUTURE WORK

Allocation of resources is a significant and visible challenge in the cloud-based environment. The effectiveness of a cloud environment is determined by load-balancing algorithms, which assign appropriate VMs to incoming tasks. This research aims to compare the contributions of several metaheuristic or swarm-based load-balancing approaches in cloud systems. The technique was tested in a CloudSim variant with a graphical interface called Cloud Analyst.

The outcomes are examined for various data centres (DCs) and user bases (UBs). When the ACO is compared to the GA, GWO, BAT, and PSO for varying DCs and UBs, the performance research shows that the ACO delivers the best Response Time as compared to all the other discussed techniques. When DC is set to 5, the ACO produces high DCPT, whereas PSO and GWO yield the same DCPT. The highest DCPT is observed in GA. The ACO creates DCPT that is nearly comparable to PSO when the quantity of DC is boosted. As a result of the preceding results, ACO could be investigated for crucial time-sensitive purposes such as patients' records information systems, infrastructure management systems etc. or applications that require rapid reaction for the given work. An interesting observation from this research is the performance of the Ant Colony Optimization (ACO) algorithm under conditions of high user retention. While ACO is known for its efficiency in terms of response time, the study reveals its potential degradation under certain high-load conditions. This finding is vital for CSPs with high user retention and consistent heavy loads, as it indicates the need for alternative solutions or hybrid algorithms that can maintain performance under such circumstances. It must be considered that convergence is another critical factor, no matter how much reduced response time is achieved through an algorithm in a particular scenario if user retention is high. The reason for the sudden degradation of ACO is its slow convergence.

On the contrary, GWO outperforms in case of a sudden spike in traffic because of its improved convergence. Due to the efficient resource utilization of BAT algorithm, it outperforms PSO in high user requests. Eventually, when the user requests increased globally, the PSO provided more optimized results in case of high user retention compared to BAT algorithm. These observations proved that PSO is providing the globally optimized solution, and BAT can experience premature convergence in case users continuously increase incoming requests. By providing empirical data on how different algorithms perform under various conditions, CSPs can optimize their resources according to the specific demands of their infrastructure and user base. It helps improve overall system efficiency, response times, and resource management.

As a result, all work can be accomplished in the future by developing a hybrid metaheuristic approach to get around the shortcomings of metaheuristic algorithms discussed in this presented performance analysis. The complexity of models and processes in cloud computing causes considerable changes in the allocation of virtual machines to new tasks. This research will be extended in the long term to include various QoS criteria like throughput, energy, cost, and latency for different network strategies. The challenges associated with load balancing, like heterogeneous endpoints, geographically distributed servers, single point of failure, virtualized migrations, management of storage and retrieval, capacity scaling, algorithms' performance, and so on, can be addressed in the future by utilizing some cutting-edge load balancing techniques; specifically, when combined with new QoS metrics and algorithm complex evaluation aspects [50]. Because cloud computing is grappling with increasing data, metaheuristic or swarm task scheduling techniques must advance in domains [51], [52] such as machine learning, artificial intelligence, IoT and blockchain.

Fault control, prevention, and workload transfer characteristics, which receive little emphasis in modern load-balancing techniques, are crucial for future studies and development. These would have been incorporated into the current system to increase its efficacy. Flexibility, another essential component of cloud technology, allows for the automated allocation and distribution of resources. As such, how resources might be utilized or planned to be released by keeping the same efficiency as a conventional criterion. Furthermore, deciding when to relocate and how much workload to monitor is still an essential research aspect. Developing evaluations in a cloud setting is frequently problematic due to the need for more accurate knowledge [75]. Future load can be calculated by looking at the current process parameters and the past demands. Therefore, it would be preferable to do studies that concentrate on load predictions. A detailed understanding of the tradeoff between the efficient utilization of the necessary hardware architecture and resource dependability is needed for a virtualized environment. Although heterogeneous load movement is a highly intriguing research field, its viability and affordability must first be established in future studies.

REFERENCES

[1] M. De Donno, K. Tange, and N. Dragoni, "Foundations and evolution of modern computing paradigms: Cloud, IoT, edge, and fog," *IEEE Access*, vol. 7, pp. 150936–150948, 2019.

[2] V. Arulkumar and N. Bhalaji, "RETRACTED ARTICLE: Performance analysis of nature inspired load balancing algorithm in cloud environment," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 3, pp. 3735–3742, Mar. 2021.

[3] 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.

[4] 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," *Electronics*, vol. 10, no. 21, p. 2718, Nov. 2021.

[5] M. Ala'anzy and M. Othman, "Load balancing and server consolidation in cloud computing environments: A meta-study," *IEEE Access*, vol. 7, pp. 141868–141887, 2019.

[6] R. Kumar and N. Agrawal, "Analysis of multi-dimensional industrial IoT (IIoT) data in edge-fog-cloud based architectural frameworks: A survey on current state and research challenges," *J. Ind. Inf. Integr.*, vol. 35, Oct. 2023, Art. no. 100504.

[7] F. S. Prity, K. A. Uddin, and N. Nath, "Exploring swarm intelligence optimization techniques for task scheduling in cloud computing: Algorithms, performance analysis, and future prospects," *Iran J. Comput. Sci.*, vol. 6, p. 1–22, 2023.

[8] A. F. S. Devaraj, M. Elhoseny, S. Dhanasekaran, E. L. Lydia, and K. Shankar, "Hybridization of firefly and improved multi-objective particle swarm optimization algorithm for energy efficient load balancing in cloud computing environments," *J. Parallel Distrib. Comput.*, vol. 142, pp. 36–45, Aug. 2020.

[9] 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, pp. 1–43, Mar. 2023.

[10] D. Syed, N. Islam, M. H. Shabbir, and S. B. Manzar, "Applications of big data in smart health systems," in *Handbook of Research on Mathematical Modeling for Smart Healthcare Systems*. Hershey, PA, USA: IGI Global, 2022, pp. 52–85.

[11] 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.

[12] 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.

[13] S. Sefati, M. Mousavinasab, and R. Z. 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, Jan. 2022.

[14] A. Gopu and N. Venkataraman, "Virtual machine placement using multi-objective bat algorithm with decomposition in distributed cloud: MOBA/D for VMP," *Int. J. Appl. Metaheuristic Comput.*, vol. 12, no. 4, pp. 62–77, Oct. 2021.

[15] 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, Sep. 2021, Art. no. 102353.

[16] 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, Aug. 2021.

[17] J. Zhou, U. K. Lilhore, T. Hai, S. Simaiya, D. N. A. Jawawi, D. Alsekait, S. Ahuja, C. Biamba, and M. Hamdi, "Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing," *J. Cloud Comput.*, vol. 12, no. 1, pp. 1–21, Jun. 2023.

[18] S. R. Jena, R. Shanmugam, K. Saini, and S. Kumar, "Cloud computing tools: Inside views and analysis," *Proc. Comput. Sci.*, vol. 173, pp. 382–391, Jan. 2020.

[19] Y. Zhang, S. Wang, and G. Ji, "A comprehensive survey on particle swarm optimization algorithm and its applications," *Math. Problems Eng.*, vol. 2015, pp. 1–38, Feb. 2015.

[20] P. Singh, M. Dutta, and N. Aggarwal, "A review of task scheduling based on meta-heuristics approach in cloud computing," *Knowl. Inf. Syst.*, vol. 52, no. 1, pp. 1–51, Jul. 2017.

[21] 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.

[22] 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.

[23] K. Geeta and V. K. Prasad, "Multi-objective cloud load-balancing with hybrid optimization," *Int. J. Comput. Appl.*, vol. 45, no. 10, pp. 611–625, Oct. 2023.

[24] G. Annie Poornima 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.

[25] S. Jain and A. K. Saxena, "A survey of load balancing challenges in cloud environment," in *Proc. Int. Conf. Syst. Model. Advancement Res. Trends (SMART)*, Nov. 2016, pp. 291–293.

[26] T. Deepa and D. Cheelu, "A comparative study of static and dynamic load balancing algorithms in cloud computing," in *Proc. Int. Conf. Energy, Commun., Data Analytics Soft Comput. (ICECDS)*, Aug. 2017, pp. 3375–3378.

[27] Z. Chaczko, V. Mahadevan, S. Aslanzadeh, and C. Mcdermid, "Availability and load balancing in cloud computing," in *Proc. Int. Conf. Comput. Softw. Model.*, Singapore, 2011, pp. 134–140.

[28] W. Wang and G. Casale, "Evaluating weighted round Robin load balancing for cloud web services," in *Proc. 16th Int. Symp. Symbolic Numeric Algorithms Scientific Comput.*, Sep. 2014, pp. 393–400.

[29] 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, Jan. 2017.

[30] A. V. Lakra and D. K. Yadav, "Multi-objective tasks scheduling algorithm for cloud computing throughput optimization," *Proc. Comput. Sci.*, vol. 48, pp. 107–113, Jan. 2015.

[31] A. Thomas, G. Krishnalal, and V. P. J. Raj, "Credit based scheduling algorithm in cloud computing environment," *Proc. Comput. Sci.*, vol. 46, pp. 913–920, Jan. 2015.

[32] S. Dam, G. Mandal, K. Dasgupta, and P. Dutta, "Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing," in *Proc. 3rd Int. Conf. Comput., Commun., Control Inf. Technol. (C3IT)*, Feb. 2015, pp. 1–7.

[33] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Service load balancing, task scheduling and transportation optimisation in cloud manufacturing by applying queuing system," *Enterprise Inf. Syst.*, vol. 13, no. 6, pp. 865–894, Jul. 2019.

[34] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing," *J. Experim. Theor. Artif. Intell.*, vol. 28, nos. 1–2, pp. 97–112, Mar. 2016.

[35] S. Ghafir, M. A. Alam, F. Siddiqui, and S. Naaz, "Load balancing in cloud computing via intelligent PSO-based feedback controller," *Sustain. Comput., Informat. Syst.*, vol. 41, Jan. 2024, Art. no. 100948.

[36] 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.

[37] A. A. S. Farrag, S. A. Mahmoud, and E. S. M. El-Horbaty, "Intelligent cloud algorithms for load balancing problems: A survey," in *Proc. IEEE 7th Int. Conf. Intell. Comput. Inf. Syst. (ICICIS)*, Dec. 2015, pp. 210–216.

[38] D. Patel, M. K. Patra, and B. Sahoo, "GWO based task allocation for load balancing in containerized cloud," in *Proc. Int. Conf. Inventive Comput. Technol. (ICICT)*, Feb. 2020, pp. 655–659.

[39] D. P. Mahato, "Grey wolf optimizer for load balancing in cloud computing," in *Research Advances in Network Technologies*. Boca Raton, FL, USA: CRC Press, 2023, pp. 205–221.

[40] Y. Fahim, H. Rahhali, M. Hanine, E. H. Benlahmar, E. H. Labriji, M. Hanoune, and A. Eddaoui, "Load balancing in cloud computing using meta-heuristic algorithm," *J. Inf. Process. Syst.*, vol. 14, no. 3, pp. 569–589, 2018.

[41] M. Sumathi, N. Vijayaraj, S. P. Raja, and M. Rajkamal, "HHO-ACO hybridized load balancing technique in cloud computing," *Int. J. Inf. Technol.*, vol. 15, no. 3, pp. 1357–1365, Mar. 2023.

[42] V. S. Kushwah, S. K. Goyal, and A. Sharma, "Maximize resource utilization using ACO in cloud computing environment for load balancing," in *Soft Computing: Theories and Applications: Proceedings of SoCTA*. Berlin, Germany: Springer, 2020, pp. 583–590.

[43] K. Kavitha and S. C. Sharma, "Performance analysis of ACO-based improved virtual machine allocation in cloud for IoT-enabled healthcare," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 21, p. e5613, Nov. 2020.

[44] B. H. Abed-alguni and N. A. Alawad, "Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments," *Appl. Soft Comput.*, vol. 102, Apr. 2021, Art. no. 107113.

[45] N. Yuvaraj, T. Karthikeyan, and K. Praghash, "An improved task allocation scheme in serverless computing using gray wolf optimization (GWO) based reinforcement learning (RIL) approach," *Wireless Pers. Commun.*, vol. 117, no. 3, pp. 2403–2421, Apr. 2021.

[46] M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M. A. Elmagzoub, G. Muhammad, and K. Hussain Talpur, "A fast converging and globally optimized approach for load balancing in cloud computing," *IEEE Access*, vol. 11, pp. 11390–11404, 2023.

[47] A. Ullah, M. H. Khan, and N. M. Nawi, "BAT algorithm used for load balancing purpose in cloud computing: An overview," *Int. J. High Perform. Comput. Netw.*, vol. 16, no. 1, pp. 43–54, 2020.

[48] H. A. Makasarwala and P. Hazari, "Using genetic algorithm for load balancing in cloud computing," in *Proc. 8th Int. Conf. Electron., Comput. Artif. Intell. (ECAI)*, Jun. 2016, pp. 1–6.

[49] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Comput. Appl.*, vol. 32, no. 6, pp. 1531–1541, Mar. 2020.

[50] S. Basu, G. Kannayaram, S. Ramasubbareddy, and C. Venkatasubbaiah, "Improved genetic algorithm for monitoring of virtual machines in cloud environment," in *Smart Intelligent Computing and Applications*. Berlin, Germany: Springer, 2019, pp. 319–326.

[51] A. Saadat and E. Masehian, "Load balancing in cloud computing using genetic algorithm and fuzzy logic," in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Dec. 2019, pp. 1435–1440.

[52] R. Gulbaz, A. B. Siddiqui, N. Anjum, A. A. Alotaibi, T. Althobaiti, and N. Ramzan, "Balancer genetic algorithm—A novel task scheduling optimization approach in cloud computing," *Appl. Sci.*, vol. 11, no. 14, p. 6244, Jul. 2021.

[53] V. Goar, M. Kuri, R. Kumar, and T. Senjyu, *Advances in Information Communication Technology and Computing*. Berlin, Germany: Springer, 2021.

[54] 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, Jan. 2019.

[55] R. Agarwal, N. Baghel, and Mohd. A. Khan, "Load balancing in cloud computing using mutation based particle swarm optimization," in *Proc. Int. Conf. Contemp. Comput. Appl. (IC3A)*, Feb. 2020, pp. 191–195.

[56] A. Rezaee Jordehi and J. Jasni, "Particle swarm optimisation for discrete optimisation problems: A review," *Artif. Intell. Rev.*, vol. 43, no. 2, pp. 243–258, Feb. 2015.

[57] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation," *Comput. Netw.*, vol. 162, Oct. 2019, Art. no. 106860.

[58] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing," *Int. J. Speech Technol.*, vol. 49, no. 9, pp. 3308–3330, Sep. 2019.

[59] Z. Miao, P. Yong, Y. Mei, Y. Quanjun, and X. Xu, "A discrete PSO-based static load balancing algorithm for distributed simulations in a cloud environment," *Future Gener. Comput. Syst.*, vol. 115, pp. 497–516, Feb. 2021.

[60] 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.

[61] A. Gupta and R. Garg, "Load balancing based task scheduling with ACO in cloud computing," in *Proc. Int. Conf. Comput. Appl. (ICCA)*, Sep. 2017, pp. 174–179.

[62] 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.

[63] J. Chételat, M. B. C. Hickey, A. J. Poulain, A. Dastoor, A. Ryjkov, D. McAlpine, K. Vanderwolf, T. S. Jung, L. Hale, E. L. Cooke, and D. Hobson, "Spatial variation of mercury bioaccumulation in bats of Canada linked to atmospheric mercury deposition," *Sci. Total Environ.*, vol. 626, pp. 668–677, Jun. 2018.

[64] B. Raj, P. Ranjan, N. Rizvi, P. Pranav, and S. Paul, "Improvised bat algorithm for load balancing-based task scheduling," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Berlin, Germany: Springer, 2018, pp. 521–530.

[65] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment," *Int. Arab J. Inf. Technol.*, vol. 17, pp. 73–81, Jan. 2020.

[66] R. Valarmathi and T. Sheela, "Ranging and tuning based particle swarm optimization with bat algorithm for task scheduling in cloud computing," *Cluster Comput.*, vol. 22, no. S5, pp. 11975–11988, Sep. 2019.

[67] M. A. Mosa, A. S. Anwar, and A. Hamouda, "A survey of multiple types of text summarization with their satellite contents based on swarm intelligence optimization algorithms," *Knowl.-Based Syst.*, vol. 163, pp. 518–532, Jan. 2019.

[68] G. Muthsamy and S. R. Chandran, "Task scheduling using artificial bee foraging optimization for load balancing in cloud data centers," *Comput. Appl. Eng. Educ.*, vol. 28, no. 4, pp. 769–778, Jul. 2020.

[69] B. N. Gohil and D. R. Patel, "A hybrid GWO-PSO algorithm for load balancing in cloud computing environment," in *Proc. 2nd Int. Conf. Green Comput. Internet Things (ICGCIoT)*, Aug. 2018, pp. 185–191.

[70] L. Xingjun, S. Zhiwei, C. Hongping, and B. O. Mohammed, "A new fuzzy-based method for load balancing in the cloud-based Internet of Things using a grey wolf optimization algorithm," *Int. J. Commun. Syst.*, vol. 33, no. 8, p. e4370, May 2020.

[71] S. Ouhame, Y. Hadi, and A. Arifullah, "A hybrid grey wolf optimizer and artificial bee colony algorithm used for improvement in resource allocation system for cloud technology," *Int. J. Online Biomed. Eng. (iJOE)*, vol. 16, no. 14, p. 4, Nov. 2020.

[72] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, Apr. 2010, pp. 446–452.

[73] V. Behal and A. Kumar, "Cloud computing: Performance analysis of load balancing algorithms in cloud heterogeneous environment," in *Proc. 5th Int. Conf. Next Gener. Inf. Technol. Summit*, Sep. 2014, pp. 200–205.

[74] S. M. Shetty and S. Shetty, "Analysis of load balancing in cloud data centers," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 1, pp. 1–9, 2019.

[75] I. Behera and S. Sobhanayak, "Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach," *J. Parallel Distrib. Comput.*, vol. 183, Jan. 2024, Art. no. 104766.

DARAKHSHAN SYED received the Master of Engineering (M.Eng.) degree in computer networking and performance evaluation from the NED University of Engineering and Technology. She is currently pursuing the Ph.D. degree with the Computer Science Department, Bahria University, Karachi. She is a Lecturer with the Computer Science Department, Bahria University. She is an Experienced Tutor of computer courses, such as programming languages, information security, artificial intelligence, software engineering, and mobile application and development. She was a Research Assistant with a demonstrated history of working in the education management industry.

GHULAM MUHAMMAD SHAIKH received the B.E. degree in computer systems and the M.E. degree in communication systems and networks from the Mehran University of Technology, Jamshoro, and the Ph.D. degree in parallel processing from the Beijing Institute of Technology, Beijing, China. He joined the Computer Science Department, Bahria University, Karachi Campus, as an Associate Professor. After the Ph.D. degree, he has published more than 25 research articles in different international and nationally recognized journals. He is an Expert in multi-discipline domains, such as cloud computing, networking, artificial intelligence, machine learning, SDN, the Internet of Things (IoT), and robotics. He conducts workshops on cloud computing.

HANI MOHAMMED ALSHAHRANI received the bachelor's degree in computer science from King Khaled University, Abha, Saudi Arabia, the master's degree in computer science from California Lutheran University, Thousand Oaks, CA, USA, and the Ph.D. degree from Oakland University, Rochester, MI, USA. Currently, he is an Associate Professor of computer science and information systems with Najran University, Najran, Saudi Arabia. His current research interests include smartphones, the IoT, crowdsourcing security, and privacy.

MOHAMMED HAMDI received the bachelor's degree in computer science from Jazan University, Jazan, Saudi Arabia, in 2009, and the Master of Science and Doctor of Philosophy (Ph.D.) degrees in computer science from Southern Illinois University, Carbondale, IL, USA, in 2013 and 2018, respectively. He is currently an Associate Professor and the Dean of the College of Computer Science and Information Systems, Najran University. His research interests include databases, query optimization, data mining, big data, and security.

MOHAMMAD ALSULAMI received the Ph.D. degree in computer science and engineering from the University of Connecticut, Storrs, CT, USA. He has been teaching graduate and undergraduate students in computer science and engineering for the past 14 years. He is currently an Assistant Professor with the College of Computer Science and Information Systems, Najran, Saudi Arabia. He has published many research articles in international journals and reputed international conferences. His current research interests include cloud computing, the IoT, machine learning, and underwater systems.

ASADULLAH SHAIKH (Senior Member, IEEE) received the B.Sc. degree in software development from the University of Huddersfield, U.K., the M.Sc. degree in software engineering and management from Goteborg University, Sweden, and the Ph.D. degree in software engineering from the University of Southern Denmark. He was a Researcher with UOC Barcelona, Spain. He is currently a Professor, the Head of research and graduate studies, and the Coordinator of seminars and training with the College of Computer Science and Information Systems, Najran University, Najran, Saudi Arabia. He has more than 140 publications in the area of software engineering in international journals and conferences. He has vast experience in teaching and research. His current research interests include UML model verification, UML class diagrams verification with OCL constraints for complex models, formal verification, and feedback techniques for unsatisfiable UML/OCL class diagrams. He is an Editor of *International Journal of Advanced Computer Systems and Software Engineering* (IJACSSE) and on the International Advisory Board of several conferences and journals. Further details can be obtained using www.asadshaikh.com.

SYED RIZWAN received the master's degree in engineering (computer science and technology) from the Harbin Institute of Technology, China. He is currently a Lecturer with the Faculty of Engineering, Science and Technology, Iqra University, Karachi, Pakistan. His goal as a Teacher is to ensure that students are well prepared to serve their society and community, especially as professional engineers or computer scientists. His research interests include machine learning, artificial intelligence, cloud computing, and blockchain technology. He possesses good skills in MATLAB, Python, and Weka.