Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

Comparative analysis of metaheuristic load balancing algorithms for efficient load balancing in cloud computing

Jincheng Zhou[1], Umesh Kumar Lilhore[2], Poongodi M[3], Tao Hai[1,4], Sarita Simaiya[2], Dayang Norhayati Abang Jawawi[4], Deemamohammed Alsekait[5], Sachin Ahuja[2], Cresantus Biamba[6*] and Mounir Hamdi[3]

Abstract

Load balancing is a serious problem in cloud computing that makes it challenging to ensure the proper functioning of services contiguous to the Quality of Service, performance assessment, and compliance to the service contract as demanded from cloud service providers (CSP) to organizations. The primary objective of load balancing is to map workloads to use computing resources that significantly improve performance. Load balancing in cloud computing falls under the class of concerns defined as "NP-hard" issues due to vast solution space. Therefore it requires more time to predict the best possible solution. Few techniques can perhaps generate an ideal solution under a polynomial period to fix these issues. In previous research, Metaheuristic based strategies have been confirmed to accomplish accurate solutions under a decent period for those kinds of issues. This paper provides a comparative analysis of various metaheuristic load balancing algorithms for cloud computing based on performance factors i.e., Makespan time, degree of imbalance, response time, data center processing time, flow time, and resource utilization. The simulation results show the performance of various Meta-heuristic Load balancing methods, based on performance factors. The Particle swarm optimization method performs better in improving makespan, flow time, throughput time, response time, and degree of imbalance.

Keywords Metaheuristic algorithms, Resource management, Load balancing, Cloud computing, Load balancing metrics

*Correspondence:
Cresantus Biamba
cresantus.biamba@hig.se
[1] School of Computer and Information and Key Laboratory of Complex Systems and Intelligent Optimization of Guizhou, Qiannan Normal University for Nationalities, Duyun, Guizhou 558000, China
[2] Department of Computer Science and Engineering, Chandigarh University, Mohali, Punjab, India
[3] College of Science and Engineering, Hamad Bin Khalifa University, Qatar Foundation, Doha, Qatar
[4] School of Computing, Faculty of Engineering, Universiti Teknologi Malaysia (UTM), Skudai, 81310 UTM Johor Bahru, Johor, Malaysia
[5] Department of Computer Science and Information Technology, Princess Nourah Bint Abdul Rahman University, Applied College, Riyadh 11564, Saudi Arabia
[6] Department of Culture Studies, Religious Studies and Educational Sciences, University of Gävle, 801 76 Gävle, Sweden

Introduction

Load balancing is vital in optimizing the utilization of cloud computing resources, i.e., processors, storage, and memory. Virtual machines running on physical machines are responsible for allocating and using resources. Some VMs may be over-used and under-used when workloads are processed on VMs. Load balancing techniques ensure that each machine in the cloud data center will perform the same number of tasks at any given time per their capacity. User demands are incredibly dynamic in cloud computing, and achieving multi-tenancy requires separating different users in the cloud infrastructure [1]. In existing cloud computing research, different heuristic and Metaheuristic methodologies were used by various researchers to distribute Load among VMs and to

achieve optimal utilization of cloud resources and better performance.

The challenge of mapping workload on massive computing resources in cloud computing relates to classifying complications known as "NP-hard" challenges. No optimization algorithm for such difficulties may generate an optimal remedy inside polynomial time. Solutions predicated on extensive review are not technically feasible as the functioning cost of producing work schedules is exceptionally high. The primary objective of the load-balancing method is to distribute the workloads among VMs and computing resources to minimize the relative imbalance [2]. In cloud computing, heuristic and Metaheuristic methods are widely used to achieve load balancing. These methods have various vital features, such as a more prominent search space with a random search that helps find an optimum solution in a fixed time for a scheduling problem.

The computational cost of the metaheuristic algorithm is higher than the heuristics algorithm. Most researchers utilize a heuristics method that reduces the search space to improve the convergence rate of metaheuristic methods. There are several objectives in this process [3].

Need for load balancing

In cloud computing, architecture workload balancing is an essential factor that helps allocate computing resources. Each VM has a different processing speed, storage capacity, and memory. Load balancing is the only way to map a workload with a perfect VM so that any VM cannot be overloaded. A Cloud model encounters request overload due to dynamic computing through the web [4]. In Cloud computing, load balancing is the most complex and essential research area for distributing workloads amongst VMs in data centers. Cloud computing mainly focuses on the principle of on-demand resource sharing using the internet. The critical components of cloud computing include interconnected computing devices, storage, and data centers [5].

A distributed and parallel computing strategy is used in cloud computing to share data, software, hardware, and computing resources with other devices. This model offers a "pay-per-use" model. The customer does not need to purchase any computational platforms or software to perform a task; a user only needs the internet to access the cloud services and computing resources and pays per service type and utilization. It reduces the cost of buying a software suite that is not needed full-time and allows for the dynamic utilization of resources that multiple users can access simultaneously without compromising service quality. Cloud service providers experience difficulties related to the quality service owing to the following reasons:

- The size and complexity of the public cloud
- The potential weaknesses of conventional load-balancing algorithms
- The variation of key stakeholders whose function is to perform customer queries

Motivation

In cloud computing, a load balancing technique evenly transfers the workload volume across all the VMs as per their capacity to achieve optimum resource utilization. Metaheuristic load-balancing methodologies covered in this research depend on multiple metrics. In a multi-cloud environment, load balancing is a difficult task. Further research has been done on multi-cloud technology to solve issues, i.e., vendor lock-in, quality, reliability, and interoperability [6].

In a multi-cloud environment, the distribution of computing resources is always challenging. Various researchers suggest different resource allocation policies to achieve optimum resource allocation. Table 1 represents a comparison of the present review and previous research.

Load balancing is essential to achieve the quality of service and optimum resource utilization in heterogeneous cloud computing. Load balancers assist in an equal and fair resource allocation to workload for optimum resource utilization and customer satisfaction at the least price. The existing load-balancing methods encounter several issues which need immediate attention. It motivates researchers to discover better load-balancing policies to overcome these difficulties [7]. Metaheuristic-orientated techniques mainly overcome these challenges by offering accurate solutions in a reasonable period. Metaheuristic load balancing has attracted increased attention in recent decades due to its performance,

Table 1 Comparison of Metaheuristic Load balancing methods (present review Vs Previous research)

Reference	Comparison based analysis	Latest State-of-art	Taxonomy	Graphical analysis
[1]	Yes	No	No	No
[2]	Yes	No	No	No
[3]	Yes	Yes	No	No
[4]	Yes	No	No	No
[5]	Yes	Yes	No	No
[7]	Yes	No	Yes	Yes
[6]	Yes	No	No	No
[8]	Yes	Yes	No	No
Present review	Yes	Yes	Yes	Yes

reliability, quality of service, and efficiency in overcoming massive and complicated challenges. In previous research, Metaheuristic based strategies have been confirmed to accomplish accurate solutions under a decent period for those kinds of issues.

A detailed review of metaheuristic methods is needed based on various factors, i.e., taxonomy, algorithms, parameters, and performance. This paper provides an extensive survey and comparative analysis of metaheuristic load-balancing algorithms for cloud and grid environments. In the available literature, there is no taxonomy to classify distinct scheduling algorithms. This research provides a comprehensive view of the state-of-the-art cloud load balancing methods. It examines Metaheuristic Load balancing algorithms, taxonomy, key features, and challenges [8]. Table 2 reviews various queries related to Metaheuristic Load balancing research.

Contribution

The existing metaheuristic load balancing surveys encounter several issues, i.e., no detail, taxonomy, no comparisons based on the current state of the art, and no key features, challenges, and architecture covered. Load balancing techniques vary depending on the dependency between many activities to be scheduled to take place. If precedence rules occur in activities, an action can only be planned once all its family activities are finished. In contrast, activities are independent in the scenario, and individuals can be scheduled in any specific order.

The methods are "dependent workflow scheduling methods" and "independent workflow scheduling methods." All these load-balancing algorithms based on metaheuristic modelling techniques are discussed in subsequent subsections. This article provides a comprehensive and systematic survey of the most recent Metaheuristic load-balancing algorithms to provide an operational understanding of these methodologies.

Paper organization

The research article is organized as follows. Introduction section covers the introduction of the research, and Background section describes the research background studies and the analysis of cloud stack holders, cloud load balancing, and policies. Taxonomy of load balancing algorithms section covers load balancing powerful taxonomy. Metaheuristic algorithms in cloud computing section covers metaheuristic algorithms in cloud computing, powerful taxonomy of Meta-Heuristic load balancing Algorithms, issues, and challenges. Result and discussion section covers the results and discussion, and finally, the last section covers the conclusion and future work of the research.

Background

Cloud computing varies significantly from other hosting alternatives due to the two key components, CSP and CSU. It is not uncommon for a service to be billed per user in the cloud computing service model, so there

Table 2 Review of Major research questions related to Load balancing

Objectives and Activities	Research Queries
Load Balancing	What are the key parameters? Why is load balancing? What are the available primary researches? How do Metaheuristics play a vital role in load balancing?
Optimum Resource Utilization	How can we achieve optimum resource utilization? How can it affect cloud performance? How can it increase efficiency?
Metaheuristic LB Methods	What are the available MLB methods? What is the various taxonomy of MLB? What are the available methods and their features, applications, and issues?
Quality of Services improvement	What are the various QoS criteria that must be identified? What are the Specifications associated with resource scheduling?
Cloud Simulation/ Implementation Tool	What simulation tools are used to implement MLB? What are the properties of a simulation model used to implement MLB? What are the characteristics provided by different innovative simulation software for MLB?
Energy Optimization	What is the role of energy optimization in cloud load balancing? What are the challenges in energy optimization? How can metaheuristic methods resolve energy optimization challenges?
Challenges in Cloud Computing research	What are the key challenges in cloud computing? How can metaheuristic methods resolve them?
Significant issues in Cloud Computing Research	What are the major issues in cloud computing?

is no long-term contract. Self-service ensures that data are provisioned without user interaction, sales calls, new service bookings, and long and complex contractual relationships, empowering customer service and helping utilization. It indicates that the procurement of a cloud service is entirely automated and essential for creating cloud services at a reasonable price [9].

As per NIST, cloud computing can be defined as: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model comprises five essential characteristics, three service, and four deployment models." (Source: NIST Cloud Definition).

Major stockholders of cloud

The Main stack holders of cloud computing include cloud end users, developers, brokers, policymakers, and service providers. Figure 1 describes the taxonomy for cloud stockholders.

- *Cloud End-user*: The end users are primary customers utilizing cloud computing services.
- *Cloud Brokers*: An object that handles cloud services' usage, efficiency, and distribution and tries to negotiate interactions between providers and customers.
- *Cloud Carrier*: The carrier cloud is an entity that combines two or more devices and some other information and communication features to help implement high-demand cloud-based services.
- *Cloud Developer*: In this field, developers create software hosted in the cloud. Developers must spend time in various phases, i.e., analyzing customer needs, problem formulation, solution system designing, coding, debugging, and deployment.
- *Cloud Service Providers*: A CSP mainly offers four cloud computing models public, private, community, and hybrid. CSP is primarily responsible for creating cloud services, maintaining the quality of service, and ensuring precise distribution. Various companies are using their private clouds just for inner usage. The cloud computing model based on cloud service providers is as follows:

  ➢ Private clouds- It is mainly related to an organization and only used by particular users.
  ➢ Public clouds- In public cloud services, i.e., platform, infrastructure, software, and data are offered by a third party, and users can access the services via the internet. Examples are Google Compute Engine, Amazon Web Services, HP Cloud, and Microsoft Azure.
  ➢ Community Cloud- It is a cloud structure that enables services and applications to be usable by a community of numerous institutions to exchange relevant data.
  ➢ Hybrid cloud- It is a composition of private and public cloud resources. It enables organizations to increase some internal tools and some infrastructure from outside. The task of "resource provisioning" must be completed by the cloud service provider.

- *Cloud Policy Makers*: Cloud policies are the rules that regulate how businesses use the cloud. It is mainly used to maintain the authenticity and confidentiality of the data. A company, organization, or government agency can be a cloud policymaker [10].

Load balancing in cloud computing

Cloud load balancing (CLB) is a process that distributes workloads and computing resources in a cloud environment. Load balancing enables organizations to handle

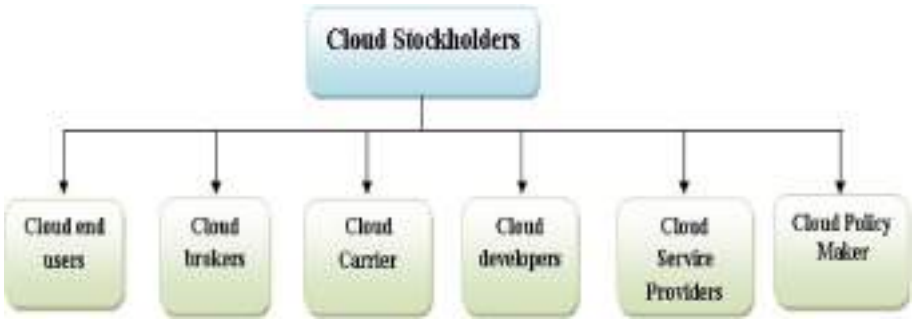

Fig. 1 Cloud Stack holders

implementation and workload requirements by distributing resources between different computing resources, i.e., Virtual Machines, storage, networks, and data centers [11].

Load balancing framework

A load balancer determines which VM can accurately handle the subsequent incoming user request without compromising the quality of service and load optimization policies. Workload management is the critical responsibility of a cloud data center control system. Workloads are routed to the load balancer, which uses a load-balancing technique to allocate work activities to the appropriate Virtual machine. A VM manager is a vital component of Virtual machines. In cloud computing, virtualization is a well-known technique.

Figure 2 describes a load-balancing architecture. When the load balancer gets customer service requests, it applies an appropriate load-balancing approach to map the recommendations with the precise VMs.The primary goal of virtualization is to share powerful machines between many VMs. A VM is a virtual computer system server on which software packages can be operated. VMs mainly handle the customer's requests. In cloud computing, environment users can be from any global location and post their requests irregularly. These user requests must be forwarded to the correct VMs for handling. Accurate workload distribution is an essential issue in cloud computing. The quality of service can be affected if any VMs become overloaded or underloaded. When a Cloud customer gets dissatisfied due to poor quality of service, they can quit the utilization of the cloud and never return [12].

Load balancing metrics

A Cloud monitoring system gathers measurement results to understand how a cloud computing model and services function. A set of these parameters is commonly used as a "metric." This subsection covers cloud computing load-balancing metrics based on existing research and load-balancing algorithms [13].

- Response time: This is the time required for the system to finish a job. The number of processes completed for every time interval is calculated using criterion.
- Makespan time determines the highest finish time or the time it takes to distribute resources toward a consumer.
- Fault tolerance: It defines the application's capabilities to accomplish load balancing throughout the occurrence of specific links and link breakdowns.
- Scalability: It refers to an application's capacity to execute homogeneous load balancing throughout the framework based on demands as the size of the network grows. The automated system of selection is highly configurable.
- Migration time: Moving an assignment from an overloaded server to an underloaded server takes time.
- Degree of imbalance: It determines how evenly VMs are distributed.

Load balancing policies

The following load-balancing policies are widely used in cloud computing [14].

- Location policy- It mainly identifies unused or underutilized VMs and then al-locates work to these VMs for reprocessing. After defining the necessary information for work migration using three methods: probing, negotiation, and a random selection, It selects the target node. The location policy sets the target randomly and transfers the work activities.

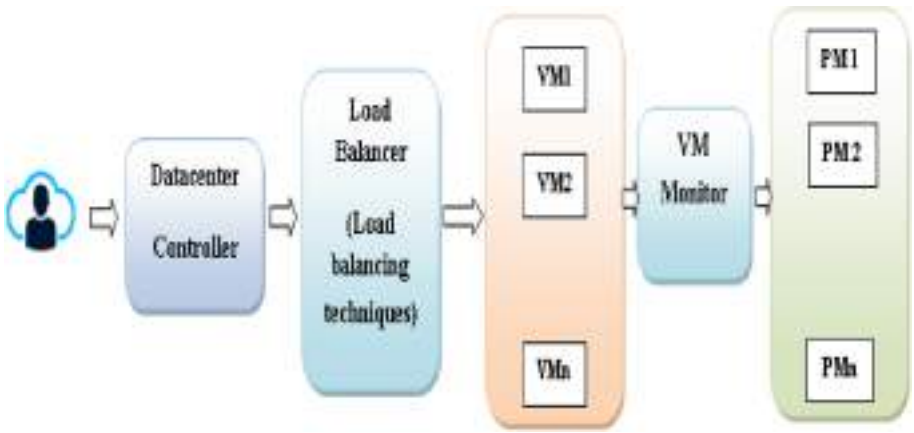

Fig. 2 Load balancing framework in Cloud computing

- Selection policy- This policy defines the work activities which can be selected and moved from one device to the next. It mainly prefers work activities, which depend on the number of features and structure for migration.
- Information policy-It is another dynamic load-balancing policy that stores all the resource data in the system, which can be used by many other approaches to take action. It sets the methods for data collection. Nodes presently gather data using the Agent technique. The supply, routine, and state change policies are examples of different information policies.
- Transfer policy identifies the conditions under which the workloads can be transferred from one network device to another target device. It uses two methods to recognize the work activities to be moved: "all recent" and "last obtained." All arriving activities enter the "last obtained" strategy, and the last action enters "all recent." The transfer policy is premised on determining whether a move can be transferred (task migration) and which function can be applied (task rescheduled).

Taxonomy of load balancing algorithms

This section represents the categorization of existing load-balancing methodologies. Load-balancing methods can be categorized into two phases: a) based on state of the art and b) based on the process initiated in the system. Each category can be further divided into static and dynamic techniques. The performance of cloud computing directly depends on the type of technique. The most popular static load balancing methods are Round Robin, Weighted Round-Robin, Min-Min, Max-Min, and Opportunistic load balancing methods. The dynamic algorithms include metaheuristic methods [15]. Figure 3 represents the primary taxonomy of cloud load balancing methods.

Load-balancing methods can be categorized into two phases: a) based on state of the art and b) based on the process initiated in the system. Each category can be further divided into static and dynamic techniques.

Load balancing depends on the system state

Load balancing methods based on system state can be divided into the following categories [16].

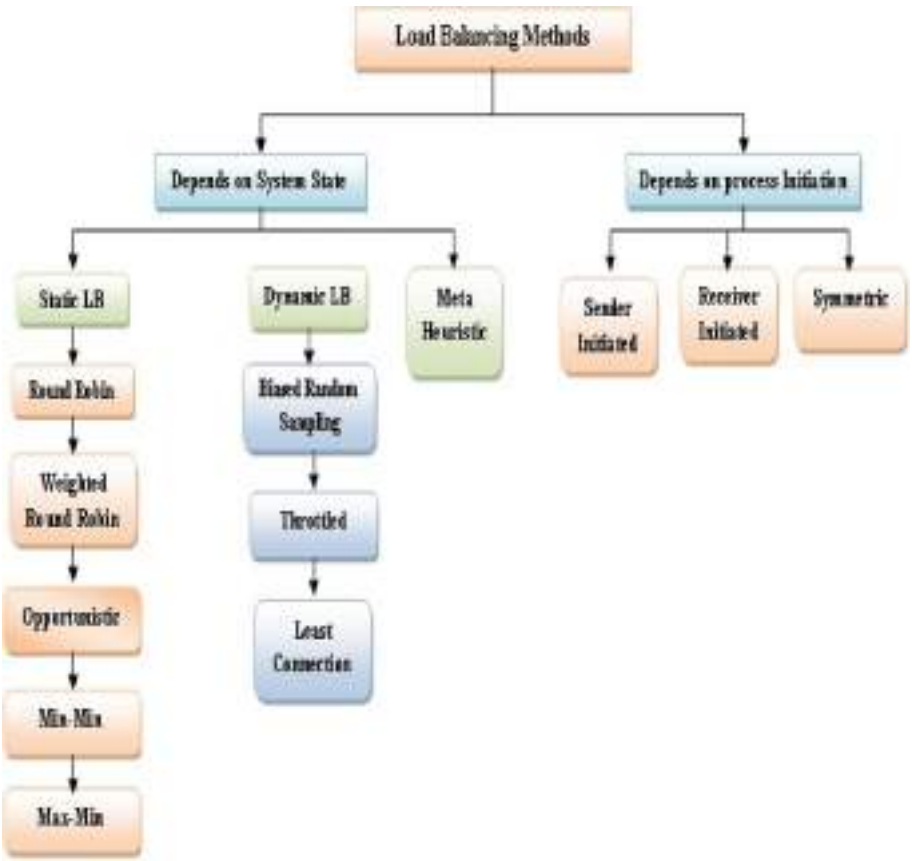

Fig. 3 Taxonomy Load balancing methods in cloud computing

Static load balancing

A static load balancing method mainly ignores the current system state. A system state contains data like the loading condition; when a static load balancing method performs load balancing, the system performance can be affected due to the overload or underload of the VM. Static load balancing procedures are primarily based on observing the system's typical behaviour; transfer choices are independent of the underlying current structure state [17].

- Optimal Load Balancing: The DCN gathers resources and other necessary information and submits work activities to the cloud load balancer, optimizing allotment in the shortest time.
- Suboptimal Load Balancing: In this technique, when the load balancer cannot determine the optimal decision, a suboptimal solution can be calculated for any problem. The primary examples of suboptimal load balancing methods are Max–Min, Min-Min, Shortest Job First, Round Robin, Central Load Balancing, and Opportunistic Load Balancing.

The most popular static load balancing methods are as follows.

Round Robin: It is one of the most specific load-balancing methods because it uses a time-triggered scheduling scheme that is very practical and reliable. In this method, time is divided into slices and quantum. This method utilizes a round-robin algorithm to allocate tasks to machines. The process selects random nodes when applying load balancing. This algorithm mainly depends on data centers. The functioning of the round-robin occurs when online consumers request the cloud system for any job process; then, this request will be assigned to the data center console and managed by a round-robin method [18].
Weighted Round Robin: This method utilizes the VMs' resources and capacities. This method mainly works on a critical principle: allocating a powerful virtual machine to an activity with more work. This method assigns a weight to each process based on its capacity. The system maintains a table to keep track of the records of the weighted list of servers. This process takes more time than the round-robin method.
Opportunistic Load Balancing (OLB): OLB is the method that allocates workflow to nodes in an available sequence. It is quick and easy and does not consider the estimated completion period of each device. It is a static load-balancing method that does not consider the existing workflow of each device. Hence it retains every server active by randomly spreading all uncompleted work activities to the available servers. It makes the method deliver disappointing results on task scheduling. It struggles to determine the node's complexity, further decreasing the processing activity's efficiency. Additionally, the cloud system will experience bottlenecks [19].
Min-Min Load balancing: This method begins with a list of activities that are not mapped. This method selects a machine with the shortest completion time for all jobs. It allocates resources to a user request that requires a minimum completion time. A table keeps the records of system state and node information. The method repeats the allocation process until all unmapped activities are assigned to a VM [20].
Min-Max Load Balancing: The Max-Min algorithm is very similar to the Min-Min algorithm. This method allocates a machine with the shortest finishing time for workloads. The task with the longest finishing time is assigned to a specific resource. Also, it updates the ready and waiting time details. The complete process repeats until all tasks are correctly mapped. The objective of this method is to minimize the time it takes for large tasks to finish. Table 3 represents the review of static load balancing methods.
Dynamic Load Balancing: Dynamic load balancing methods are those methods that hunt for the lowest virtual machine in the system and then appoint a suitable massive amount upon this. This method allocates the task to all the machines at the application level. Table 4 represents the review of dynamic

Table 3 Review of static load balancing methods

References	Key methods	Benefits	Challenges
[5]	Round Robin Load Balancing	Easy to use, and higher accuracy	Higher energy consumption and takes more time
[7]	Weighted Round Robin	Permit the cloud to distribute the load unevenly	It is determined by the design and CPU usage of a Virtual machine
[6]	Opportunistic Load balancing	Provides better accuracy	It ignores each node's predicted completion time
[8]	Min-Min Load balancing	It successfully utilizes underutilized resources	Consumes more energy
[9]	Min–Max Load balancing	Minimizes the load on an overloaded system	Higher energy consumption

Table 4 Review of dynamic load balancing methods

References	Key methods	Benefits	Challenges
[11]	Biased Random sampling method	Provides better precision, and resource utilization	Consumes more energy
[12]	Throttled Load balancing method	Better resource utilization	Performs better for a similar workload environment
[13]	Least Connection method	Better accuracy and performance	Takes more time
[14]	Load balancing Depends on the Initiation process	It provides three categories: Sender, receiver, and Symmetric Load balancing	Response time is poor for higher load

load balancing methods [21]. The few dynamic loads balancing methods are as follows:

Biased Random sampling method: It is a dynamic load-balancing method; that mainly applies random sampling among all the nodes. Servers are treated as nodes. This technique is defined by a virtual graph built using interconnection to describe the load on every node. In this graph structure, each node is treated as a vertex. When a client sends a service request to the cloud, the load balancer maps the available correct VM to the user request.

Throttled method: The balancer keeps index metrics of VMs (processing speed, capacity, storage) and their current states (free of busyness). A client computer initially sends a requisition to the cloud data center to select the most appropriate VM to accomplish the preferred task.

Least Connection method: The 'Least Connections' load balancing method transfers the load by selecting the server with the lowest energetic transaction data. A dynamic scheduling method transmits user requests to the cloud server with the lowest quantity of active links when the user requisition is requested [22].

Load balancing depends on the initiation process

The following methods depend on the system Initiation process [23].

- *Sender Initiated load balancing*: An overloaded machine allocates workload in sender-initiated techniques. A Server (sender) tries to transfer work to an under-loaded server (receiver).
- *Receiver Initiated load balancing*: The load-distributing function in receiver-initiated methodologies is started by an under-loaded server (recipient) attempting to obtain work from an overloaded server (sender).
- *Symmetric load balancing*: Sender-and receiver-initiated processes are combined and applied.

Metaheuristic algorithms in cloud computing

The conventional load balancing methods are simple but do not work for more severe uncertainty problems, so metaheuristic methods are used. It is a heuristic algorithm that does not depend on the level of the problem. A metaheuristic method can be defined as an interactive formation procedure that guides the exploration process and the employ of the search space. Meta-heuristic methods are one of the methods that can be utilized to handle performance issues, including task scheduling [24]. A Metaheuristic method can be divided into two main categories: a) based on a local search and b) based on a random search.

Need for meta-heuristic algorithms

A metaheuristic technique assists in optimizing an objective function. It can be in-corporate to solve various optimization issues; Load balancing is one of them. This subsection covers the need for metaheuristic methods.

Heuristic: A heuristic method addresses a challenge more quickly and conveniently than conventional methods by compromising optimum solution, precision, accuracy, or speed. Heuristic methods are primarily utilized to rectify NP-complete difficulties, a class of complex situations [25].

- Its design is typically problem-focused.
- It is straightforward to get stuck at local optima.

Metaheuristic: These methods are similar to the heuristic approach. These techniques rely on two distinct features. The first characterization is the quantity of potential practical solutions utilized in each recursive call. We initiate with a standard preliminary solution, and for each phase of the hunt, the answer is interchanged with others. Due to the following reasons, a metaheuristic is required:

– A meta-heuristic method is suitable for a wide range of challenges.
–Suitable for Multimodal Optimization problems.
–Acceptable to discontinuous, nonlinear functions.

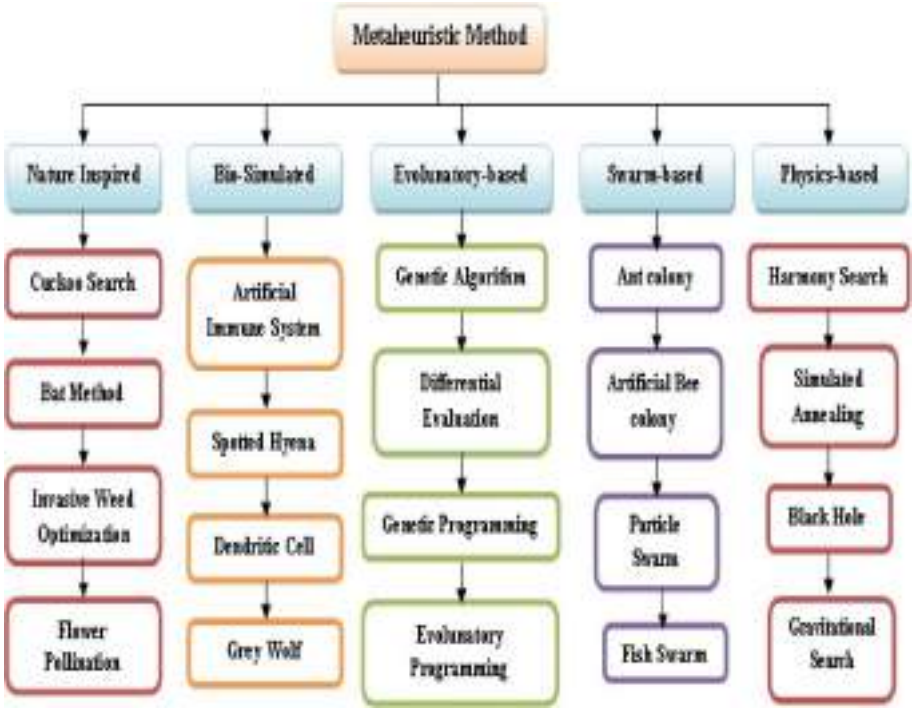

Fig. 4 Taxonomy of Metaheuristic Cloud Load Balancing

Critical elements of meta-heuristic algorithms

The following elements are mainly related to a Meta-heuristic algorithm [26].

- Exploiting Intensification: Choosing the most appropriate solution inside the existing neighbourhood. It aids in the convergence process.
- Exploration or Diversification: It is a process to find the best solution for an optimization problem by using a random sample. It keeps the process from getting stuck in local optima and increases the diversity of each key. An excellent meta-heuristic algorithm needs a good mixture of these elements to obtain the optimum result.

Main features of meta-heuristic algorithms

Meta-Heuristic Algorithms have the following key features [27].

- Nature is its primary source of inspiration.
- It is based on science, biology, and evolutionary biology principles.
- It mainly utilizes stochastic elements.
- It also involves the utilization of random factors.
- There is no restriction on using the "Hessian matrix" or "gradient."
- It utilizes different variables to solve an optimum issue.

Taxonomy of meta-heuristic load balancing algorithms

Figure 4 describes the primary taxonomy of Meta-Heuristic load balancing algorithms.

The following types of Meta-heuristic algorithms are widely used in cloud computing.

Table 5 Review of Nature Inspired Load balancing methods

References	Key methods	Research Findings	Challenges
[20]	Cuckoo Method	It generates an optimum solution	It runs into difficulty with weaker convergence
[21]	Bat Method	Enhances the cloud-computing resource distribution process	It can work better for Intra workload
[22]	Invasive Weed Optimization	Provides the solution for workflow scheduling based on multi-objective challenges	Performs better for the static workload
[23]	Flower Pollination	It shows better results for NP-Complete issues	Encounter with overflow

*Nature-inspired Algorithms*It is a sequence of unique problem-solving approaches and techniques obtained from natural operations and frequently utilized to solve numerous optimization challenges. Table 5 represents the review of various nature-inspired load balancing methods. The type of nature-inspired algorithms in cloud computing are as follows-

Cuckoo Search Algorithm (CSA): The CSA method is based on the behaviours of "cuckoo birds." Cuckoos are lovely birds, but their violent and aggressive reproduction tactic impresses everybody. Cuckoos lay their eggs inside large communal nests. Female cuckoos recreate the colours and shapes of microbial enzyme eggs. Cuckoos decrease the risks of destroying eggs and enhance their productivity [28].

Input: Problem space with n numbers
Output: Output with an optimal solution

1. Let f(x) be an optimization function, with x=(x1,..., xd);
2. While (number of iteration < Maximum Generation) or (end criterion),
   i. Start with a population of n nests xi (where I =1,2,...,n).
   ii. Obtain a cuckoo (assume I) at random, then
   iii. Use Levy flights to develop a random solution;
   iv. Examine its quality and suitability; Fi
   v. Select a nest at irregular intervals from n (assume j);
   vi. if (Fi > Fj), start replacing j with the latest solution;
   vii. end
   viii. Enable a fraction (Pa) of the worst nests to be abandoned
   ix. As well as new products to be built in new areas via Levy flights;
   x. Take the best options;
   xi. Sort the options and choose the best one;
3. End while
4. Wait for the results of the post-processing and perhaps visualization;

Algorithm 1 The Cuckoo Search Algorithm's pseudo code

Bat method

A nature-based meta-heuristic widely used in global optimization problems. These methods are motivated by the echolocation habits of mini BATs with pulse and noise levels. There are nearly 1,000 species of BAT, ranging in size from 1.5 mg to more. Mini BATs usually use echolocation. They have impaired vision but excellent hearing capacity, which allows them to fly. They also utilize an echolocation method to find insects at nighttime [29].

Input: Problem space with n numbers
Output: Output with an optimal solution

Initialization: Let initial population Pxi, Pulse Prxi, Velocity Vxi, frequency Fxi, Loudness Lxi
Begin
Set the maximum number of iterations and repeat till the max
While (Current_iteration < max)
Obtained a solution by changing the position, velocity, and frequency
If (random > Prxi)
Choose an optimal solution from all available solutions
Produce a temporary solution from the optimal solution
End if,
If (random < Loudness Lxi) && (F(Pxi) < F(Px*))
Accept the new solution and increment Prxi and Lxi
End if,
Allocates ranking to all the Bats and obtained an optimal solution, Pxi
End while
Post-process the outcomes
End

Algorithm 2 The Bad Algorithm pseudo-code for cloud load balancing

Invasive Weed Optimization (IWO) method

It is a population-based natural evolutionary optimization technique influenced by the attitude of the weed colony swarm. IWO is a constant, deterministic mathematical technique reproducing weeds' colonization behaviour patterns. Initially, a workforce of preliminary seeds is randomized and distributed over the complete solution space. These weeds will eventually mature and carry out the process of the algorithms [30].

Input: Problem space with n numbers
Output: Output with an optimal solution

Generate a random population of optimum solution (Woi);
For iteration =1, to the maximum number of possible iterations (Maxitr);
Evaluate the objective function for each population;
Find out the best and lowest fitness value for each population;
For each population, perform;
Calculate the number of seeds for each individual based on the fitness value;
Randomly allocates all the discovered seeds in the complete search space by using an average allocation around the root (parent) weed;
Add all the discovered seeds to the optimum solution set;
If ((Woi== N)> Pmax)
Sort all the populations into the descending manner based on fitness value;
Truncate the population value of weeds with the lowest fitness
Until(N== Pmax)
12. End

Algorithm 3 Pseudo code for Invasive Weed Optimization

Flower pollination algorithm

It is one of the most advanced Nature-Inspired algorithms based on the biological function of pollination. The method begins by creating a specified number of participants (N), where each partition contains a set of improved factors using the optimal solution. It utilizes an indexing strategy called "flower constancy" to determine how each population's factor reduces the optimization process. The population queuing up based on flower consistency and performance [31].

```
Input:  Problem space with n numbers
Output: Output with the optimal solution
Set an objective function for optimization (minimum or
maximum) ;
f(xi)       (where i= 1,2,3,.....n);
 Initialization of Population-based on flowers/pollen gametes
 and assigned with random solutions;
Find out the optimum/best solution G* from the initial
population results;
            Modify the probability for assignment Pro (0,1);
            while (t<MaxGen )
            for i=1; n for all n flowers population)
            if (Rand  < P) ,
            Plot a step vector; which is based on a levy
distribution method
            Global pollination calculation by using the formula
x_i^(t=1)=x_i^t+ U(x_j^t-x_k^t  )
            Else if
            Try for a new optimum solution
            If the next solution is optimum, update the solution
in the previous solution
            End for
            Calculate the new best solution Bbessol
            End While
```

Algorithm 4 Pseudo Code for Flower Pollination Algorithm

Bio-simulated algorithm

These methods are influenced by the biological behaviour patterns of animals or birds. They are mainly used to search for the optimum solution. Table 6 represents the review of various bio-simulated load balancing methods.

– *Artificial Immune System (AIS)*: The natural system is an advanced biological and autonomic nervous system that protects itself by becoming highly distributed, reliable, flexible, and self-organizing. This process can classify all new cells and particles inside the body. AIS techniques are a novel evolving intelligence strategy influenced by immunology. These processes invest in the reliable computational power of biological ecosystems like pattern recognition, extraction of features, memory, learning, diversity, distributive nature, and multi-layered protection, which provides the capabilities to accomplish numerous complex optimization problems in a highly distributed and parallel manner.

```
Input:  Problem space with n numbers
Output: Output with an optimal solution
1.      Define a threshold value Th of virtual
machine clusters for Cn clients
2.      Begin
3.      Let Antigen Agn,
4.      For each Antigen Agn,
5.      For each virtual machine cluster for Cn
i.      Affinity (Agn,  Cn)
6.      End For
7.      Find the cluster Cn, with an optimum
Affinity of  AFM
8.      If (Affinity(Agn, Cn) than
9.      Update the new Affinity solution Asol to
Cn by reproducing the Agn
10.     Else
11.     Generate the new Cn and update the new
Asol to Cn by regenerating the Agn
12.     End if
13.     End For
14.     End
```

Algorithm 5 Pseudo Code for Artificial Immune System

Spotted Hyena Optimization (SHO) Method

SHO is a brilliant technique influenced by the biological behaviour of hyenas. The SHO technique uses four stages based on the spotted Hyena's natural habit. The behaviour patterns involve a) hunting prey phase, b) searching prey phase, c) encircling prey phase, and d) attacking prey.

Table 6 Review of Bio Simulated Load balancing methods

References	Key methods	Research Findings	Challenges
[24]	Artificial Immune system	Generates an effective optimization result	Encounters various security challenges
[25]	Spotted Hyena	higher rate of integration	Inadequate Allocation for Jobs
[26]	Dendritic Cell	It helps in the identification of secure and correct nodes	It performs better for Intra jobs
[27]	Grey Wolf	For a constrained load, function best in cloud job scheduling	Encounters with overloading issues

Input: Run population initialization as Pop with i = 1, 2,,… n.
Output: best search agent is the result.

1. Initialize the parameters R, K, L, and N.
2. Assess the objective function.
3. The optimal idea or search operator is pkvhy.
4. Cklhy denotes the collection of all far efficient outcomes.
5. While (Opt < Optmax) for every iteration of the algorithm pdate the workaround pv(Opt + 1) = [Cklhy / N],also the factors R, K, L, and N are modified.
6. Verify to see if any solutions extend further,
7. then the solution area and maintain them if this occurs,
8. Determine the optimal solution of each candidate solution,
9. When an improved solution
10. then the past one becomes available,
11. update the details pkvhy.
12. Cklhy should be updated concerning pkvhy.
13. Opt = Opt + 1end while return pkvhy

Algorithm 6 Algorithm for spotted Hyena

Evolunatory based algorithm

These algorithms are population-based metaheuristic optimization algorithms inspired by evolutionary computation methods. Evolutionary methods mainly utilize processes derived from natural evolvement, including selection and recombination. Table 7 represents various Evolunatory-based load-balancing methods. The types of Evolunatory load balancing methods are as follows.

– Genetic Algorithm: A genetic technique is a search heuristic method influenced by Charles Darwin's principle of natural biological evolution. This technique represents the process of natural classification in which the healthiest participants are chosen for propagation in sequence to develop offspring of the coming generation. The critical attribute of the generic method contains a) the crossover phase, b) the mutation phase, and c) the selection phase.

Input: Problem with n spaces
Output: the optimal solution will be the outcome

1. Start the generation process- Pk = a
2. number of individuals of n completely random individual citizens; 0: k:= 0
3. Calculate fitness(i) for each I Pk; do / Generate creation k + 1:
4. Make copies: Take (1) n participants and paste them into the Pk+1.
5. Crossover: Choose n Pk participants, associate them up,
6. Generate offspring, and then incorporate the offspring into the Pk+1.
7. Mutation: Pick n participants of Pk+1 and
8. Rearrange a random selection bit in each of them.
9. Calculate fitness(i) for every I Pk; / Enhance: k := k + 1;
10. Return the best fitness participant from Pk
11. if the fitness of the fittest participant in Pk is not significant.

Algorithm 7 Genetic Algorithms (n, χ, μ)

Differential Evaluation (DE): It is a meta-heuristic method based on population. It mainly improves an optimal solution via a process of evolution. This technique makes few presumptions about fundamental optimal solutions and rapidly discovers large development zones. DE is a population-dependent and feature optimization method that enhances differences between individuals. It mainly develops a community of NP-hard problems to find the best solution. In concisely, DE keeps repeating crossover, mutation, and selection operations upon the initial condition. The DE method generates a path variable and chooses feature vectors with the best fitness value for a particular problem [32].

Table 7 Review of Evolunatory Load balancing methods

References	Key methods	Research Findings	Challenges
[28]	Genetic Algorithm	It takes a long time to complete	Perform better for limited load
[29]	Differential Evaluation	Diverse optimization issues	Encounter with load distribution issues
[30]	Genetic Programming	Enables multi-optimizations to reduce average completion cost and time	Poor results for dynamic load
[32]	Evolunatory Programming	Well-suited to higher-dimensional challenges	Encounters with overflow consume more energy

Genetic Programming (GP): GP is a subfield of Machine Learning methods that use Evolutionary Algorithms. EAs are used to find specific ways to solve complications that individuals cannot overcome. It is a method of emerging applications, beginning from a massive population of unworthy (random selection) software, fit comfortably for a particular activity by implementing processes comparable to genetic approaches to the workforce of applications. The popular types of GP include a) Grammatical Evolution, b) Stack-based Genetic Programming, c) Tree-based Genetic Programming, d) Linear Genetic Programming, e) Cartesian Genetic Programming, and f) Extended Compact Genetic Programming methods.

Evolunatory Programming (EP) is among the four main evolutionary computation frameworks. It is comparable to genetic programming; however, the framework of the system to be evaluated is wholly fixed, whereas its statistical characteristics are accepted to expand.

Swarm Intelligence (SI) Algorithm: SI method is the collaborative attitude of distributed, self-organized schemes, naturally or artificially. SI methods typically form a community of autonomous agents engaging natively with each other and through their living environment. Motivation often emerges from nature, particularly biological structures. The operators follow elementary principles. There is neither a central management framework dictating how agents must act locally. SI methods mainly include Particle Swarm Optimization (PSO), Differential Evolution (DE), Ant Colony Optimization (ACO), Artificial Bee Colony (ABC), Genetic Algorithms (GA), Cuckoo Search Algorithm (CSA), and Glowworm Swarm Optimization (GSO). Table 8 represents various nature-inspired load-balancing methods [33].

– *Ant colony optimization (ACO)*: ACO is a probability method for resolving difficulties that must be restricted to getting suitable pathways via visualizations. Multi-agent methodologies influenced by the behaviour of ants are known as the "Ant Colony Optimization" method. Ants utilize swarm interaction as their primary method based on biology. The hybrid form of an ant colony and meta-heuristic search methods is perfect for various optimal solution problems in existing research [34].

```
Input:  Problem with n spaces
Output: the optimal solution will be the outcome
1. Initialization of variables
2. while (Current_Solution !=
Optimization_Solution) repeat steps 2.1-2.2
        2.1Create Ant_Colony_results();
        2.2 Calculate Ant_Pheromone(); and Set
Ant_actions();
 end while
3. Find the Ant optimization solutions
4. Set initial population Ant(i)=1,2,.....n(k), set t=0
and Start node= st, destination node= dest
5. Find solution_best(t)= 0; Set
Shortest_Path_initilization= {0}
6. Set the Newbest(t)= 0; Set Shortest_Path_length=
{0}
7. Set the ant memory=mem
8. While (st! =dest) Repeat steps
9. Solve for all the remaining nodes
(i)=1,2,.....n(k),
   Start (k) = Start of the ant node (k)      ;
        i= Start of the ant node k; repeat steps
10. construct the neighbor ant for all x using M;
11. Choose the next best node, set the resulting
probability P(xy), and set y to the new resulting
SK(t)
12. Once the (x!==y) repeat
13. Calculate the optimum results
14. END
```

Algorithm 8 Ant Colony Optimization Method

– *Artificial Bee Colony (ABC) Optimization Method*: The ABC method is a load-balancing method of searching. The ABC has been developed depending on the insects searching for food behaviour and the attitude of honeybees. Honey bees are domestic or social flying insects in the environment. ABC is a well-managed group that relies on sweetness for its power production. The bees play multiple roles

Table 8 Review of Swarm-Based Load balancing methods

References	Key methods	Research Findings	Challenges
[32]	Ant Colony Based	It achieves less makespan and better task scheduling	Unable to provide two-way load balancing
[33]	Artificial Bee Colony	Best for dynamic load, better accuracy	Encounters with task mapping issues
[34]	Particle Swarm	Best for dynamic jobs	In high-dimensional spaces, this is simple to fall toward a locally optimal
[35]	Fish Swarm	Achieve better availability and improve reliability	Its iterative methods encounter a poor convergence rate

inside the colony, including raising children and the youths, preserving the nest, and accumulating nectar [35]. Participants search for a better food supply, choosing this from several hosts while keeping precision and agility in mind. Bees are primarily split into two groups:

- Scot/Employee Bees: These employees go out on a whim to seek fresh floral spots. Once found the food sources, they returned to the colony. They conducted a dance called "Waggle" to inform the forager colonies.
- Forager Bees: These bees obey the scout bees here to the food source and start collecting honey. The hunter-gatherers may perform a waggle to entice many bees to follow them to significant food clusters.

Input: Problem with n spaces
Output: the optimal solution will be the outcome

1. Initialization of variable I = 0; and Max Ir = 0; where Max Ir equals the number of iterations and Fitness Value(i) = 0;
2. Set bees, bi = 1,2,,N; 2:
3. Determine the starting community;
4. Analyze the population's wellness;
5. Organize the candidate solution according to the wellness consequence;
6. increment the value Max Ir =Max Ir +1;
7. Repeat the same steps while ((I <= Max Ir) or (Fitness Value(i)−Fitness Value(i−1) ≤ Error))
7.1 Increase i++;
7.2 For the neighborhood search, choose between members and non-patches;
7.3 Assign forager bees to specific patches;
7.4 Analyze each patch's optimal solution;
7.5 Out of each patch, choose the fitness values bee;
7.6 Allocate the left alive bees to look for the elite patch at irregular intervals;
7.7 Calculate the quality of the elite patch in terms of fitness;
7.8 end of while loop

Algorithm 9 Artificial Bee colony method

Particle Swarm Optimizations

The PSO method is a population-based optimal solution influenced by flocking and training fish behaviour. PSO is a choice of bio-inspired techniques, and it is a simplistic individual to seek an ideal solution in the candidate solutions. It is distinct from other evolutionary algorithms, so the optimal solution is required. It is not entirely reliant on the differential form of the desired outcome. It also has a hugely few parameters [36–38]. Table 9 represents the review of Swarm-based load balancing methods based on the Simulator used.

Input: Problem with n spaces
Output: the optimal solution will be the outcome

1. Setup is the first step.
2. Define the particle's place with an evenly distributed allocation as Pi(0)U(LB, UB), in which LB and UB demonstrate the lower and upper boundaries of the solution space, respectively.
3. Set pbest to its original position: pbest(i, 0) = Pi (0).
4. Set gbest to the swarm's minimum value: gbest(0) = argminf[Pi(0)].
5. Set the velocity to Vi U(|UB LB|, |UB LB|).
6. Continue until one of the termination criteria is satisfied.
7. Do (a) Selection random numbers: r1, r2 U for each particle I = 1,..., NP (0, 1).
8. Recalculate the particle's velocity.
9. Upgrade the location of an object.
10. Upgrade the probably the most famous location of the particles.
11. pbest(i, t) = Pi(t) if f[Pi(t)] f[pbest(i, t)].
12. Notify the swarm's highest profile position if f[Pi(t)] f[gbest(t)]: gbest(t) = Pi(t).
13. t (t + 1);
14. Production gbest(t) contains the optimal solution discovered.

Algorithm 10 Particle Swarm Optimization Method

Table 9 Review of Swarm-Based Load balancing methods based on the Simulator used

Reference	Simulator/Software	Key Findings	Accuracy
[30]	Cloud Analyst	Single Objective	Average
[31]	Cloud Sim	Multiple Objectives	Better
[32]	JAVA	Multiple Objectives	Poor
[33]	Cloud Sim	Multiple Objectives	Better
[34]	Green Cloud	Multiple Objectives	Average
[35]	VM Ware and JAVA	Multiple Objectives	Better
[36]	MATLAB	Single Objective	Poor
[37]	Cloud Sim	Multiple Objectives	Better
[38]	Cloud Analyst	Multiple Objectives	Average
[39]	Work Flow-sim	Multiple Objectives	Better
[40]	MATLAB and CPP	Single Objective	Poor
[41]	Cloud Sim	Multiple Objectives	Better
[42]	JAVA	Multiple Objectives	Poor
[43]	VM Ware and JAVA	Multiple Objectives	Better
[44]	MATLAB	Multiple Objectives	Better

Performance measuring parameters

The performance of the load balancing method is measured by performance metrics parameters, including [39].

- Degree of Imbalance (DI): It specifies how much load is distributed among various VMs based on their operational capability. It is determined by the equation below.

$$DI = \frac{(MaxTime - MinTime)}{AverageTime} \quad (1)$$

- Makespan Time: It demonstrates the completion time of the recent job when all activities are planned. It is determined by the equation below.

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

- Flow Time: It is the sum of completing times of all the jobs once all activities are assigned. It is determined by the equation below.

$$\text{FlowTime} = \sum_{i=0}^{n}\text{CompletionTime} \quad (3)$$

- Response Time: This is the time to react to the scheduling algorithms. It is also the difference between the time required to complete a job and the time necessary to submit it. It is determined by the equation below.
- Resource Utilization (RU): Maintain resources as feasible once all activities are planned. It is determined by the equation below.

$$\text{RU} = \sum_{k=0}^{n}\frac{\textit{Completion Time of all} \text{ the jobs}}{(\text{Makespan Time} * \text{ Number of jobs})} \quad (4)$$

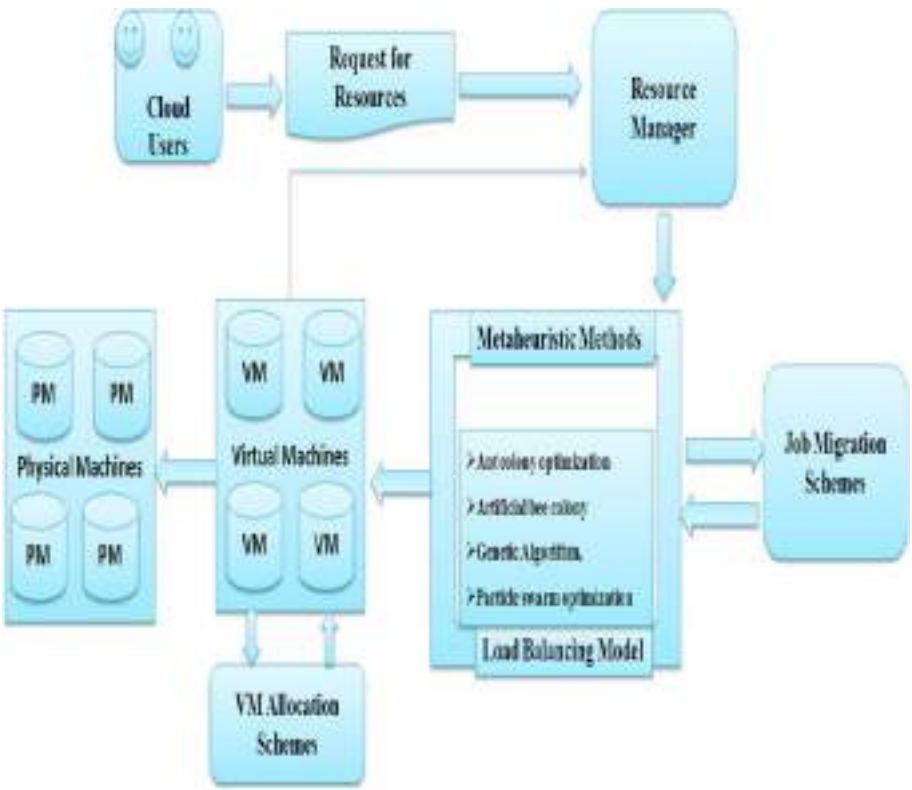

Fig. 5 Working of Metaheuristic Load Balancing Methods

Table 10 Simulation parameters used in cloud sim

Object	Parameter	Values
Virtual Machine	No of VMs	10
	Policy type	Time-sharing based policy
	Primary Memory (RAM)	1 GB
	Bandwidth	1024 MB/seconds
	Storage	50 GB
	VMM Type	Xen
	Operating system type	Linux
	Number of CPUs	Single CPU
Data Center	Number of DC	Single DC
Cloud Let	Number of Cloud lets	50–500
	Length	25,000
Cloud User	Number of users	1

Results and discussion

This section presents the comparative analysis of a few popular Meta-heuristic load balancing methods, i.e., Ant colony optimization, artificial bee colony, Genetic algorithm, and Particle swarm optimization method. These methods were implemented in a cloud sim-simulator using JAVA programming [40]. In this article, an execution time per activity depends on the job's length and VM configuration of VMs. The job size is measured in Million Instructions (MI), and the VM computational capacity is measured in Millions of Instructions per second (MIPS). Figure 5 shows the working of the proposed Metaheuristic Lead balancing model. Table 10 represents the simulation parameters used in cloud-sim [41–45].

Figures 6 and 7 show the simulation outcomes of Metaheuristic load balancing methods. The simulation is performed on the JAVA Netbeans simulator for Ant colony optimization, artificial bee colony, Genetic algorithm, and Particle swarm optimization method. Under the symmetric environment, the Meta-heuristic load balancing methods are implemented as a core component of the cloud broker. Various performance measuring parameters are calculated.

Figure 8 shows the simulation results for Response time and Data Center Processing Time outcome for metaheuristic load balancing methods.

The Ant colony optimization offers an Overall Response Time of 300.06 on average, 237.06 for Min and 369.12 seconds for max. Similarly, its shows a data processing time of 241 seconds. Another Artificial bee colony optimization method shows 278.96 seconds for overall response time and 158 seconds for data center processing time. The genetic algorithm shows 228.66 Overall response time and 146 seconds for data center processing time. PSO method shows 244.5 seconds for overall response time and 155 seconds for data center processing time.

Figure 9 shows the MakeSpan time outcomes for various metaheuristic methods. The graph is plotted among several iterations and Makespan time in seconds. The ant colony optimization method shows 17500 seconds makespan time for 200 iterations. However, Genetic algorithms show 16007 seconds which is better than other methods. For iterations 800 and 1000, all the methods show constant outcomes.

Figure 10 shows the flow time outcomes for various Meta-heuristic load balancing methods. The graph is plotted among the number of cloudlets and flow time (seconds). For 100 cloudlets Genetic Algorithm takes 151230 seconds, which is higher than the Ant colony optimization method takes 104100 seconds, and the artificial bee colony method takes. 115600 seconds and the Particle swarm optimization method takes 132450 seconds. Similar to Cloudlet 200 to 800 Genetic algorithm shows a higher flow time than other metaheuristic methods.

Figure 11 shows the imbalance outcomes for various metaheuristic load balancing methods. The graph is plotted among the number of cloudlets and the degree of imbalance. The simulationAnt-colony optimization method shows 0.234 degrees of imbalance which is the lowest degree compared to all the other metaheuristic load balancing methods. The Particle swarm optimization method offers 0.411 degrees of imbalance for 100 cloudlets. For 800 cloudlets Artificial bee colony method shows 0.8475, the ant colony optimization method 0.789, the Genetic Algorithm 0.884, and the Particle swarm optimization method offers 0.745 degrees of imbalance.

Figure 12 shows the resource utilization outcomes of various metaheuristic load balancing methods. The graph is plotted among the number of cloudlets and resource utilization %. Once the number of cloudlets increases from 100 to 800, the resource utilization % also increases. The Artificial bee colony method utilizes 30% resources for 100 cloudlets which is the highest compared to other metaheuristic load balancing methods. For 800 Cloudlets Particle swarm optimization method uses 80% of resources, and the artificial bee colony method utilizes 75% of resources, which is lesser in this category.

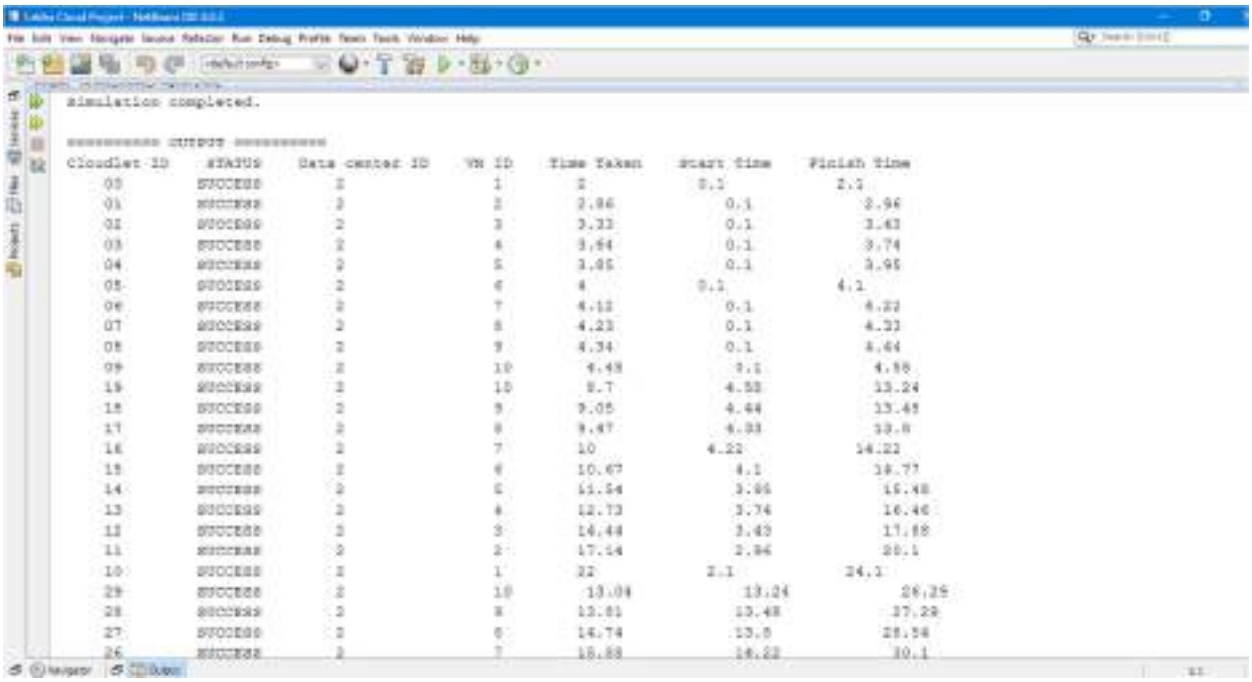

Fig. 6 Simulation outcomes for Metaheuristic load balancing methods

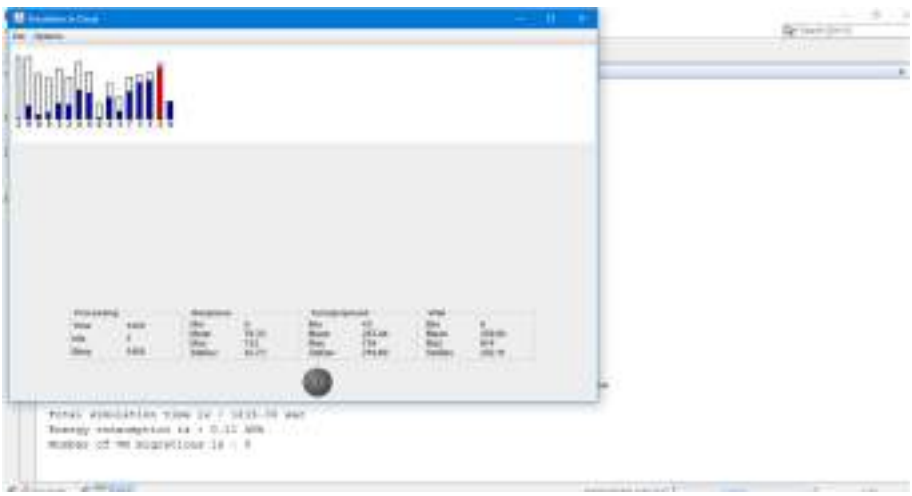

Fig. 7 Job scheduler Simulation in cloud sim for metaheuristic load balancing methods

Conclusion and future scope

The article broadly explores the application of metaheuristic methods in load balancing in cloud computing. Metaheuristic methods are particularly sluggish than evolutionary optimization techniques. The derived solutions may not be approximate solutions. Thus, plenty of investigation is toward improving the integration level and efficiency of the potential solution. Such challenges have been explored by reconfiguring the transformation operator, extracting features from the input workforce, and adopting a hybrid model in metaheuristic methods. Also, we cover various load-balancing ways focused on diversified performance parameters. Many researchers have concentrated on substantially reducing end-to-end delay and performance costs in the literature.

In contrast, others have emphasized accuracy, response time, usable capacity, processing times, and

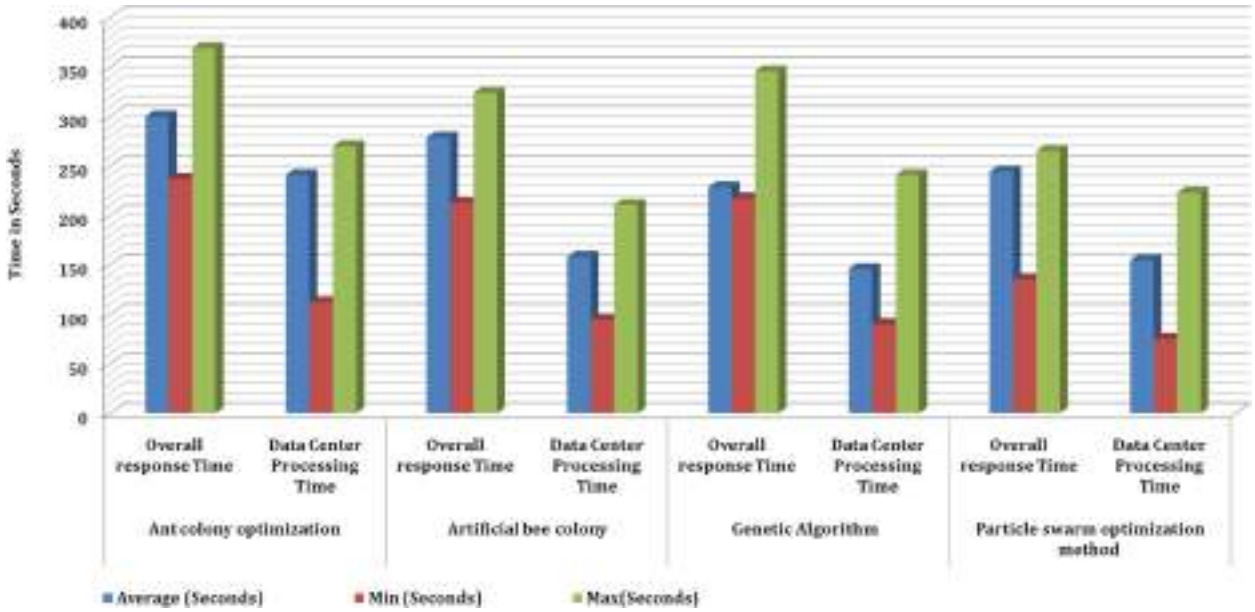

Fig. 8 Response time and Data Center Processing Time outcome for metaheuristic load balancing methods

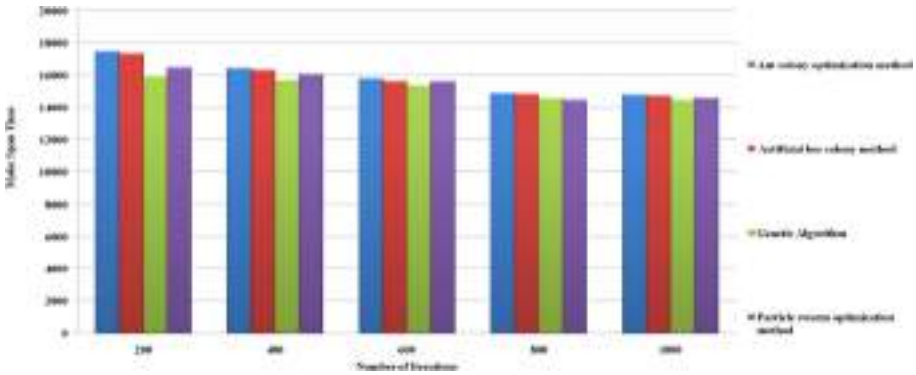

Fig. 9 Makespan Time (Sec) outcomes for metaheuristic load balancing methods

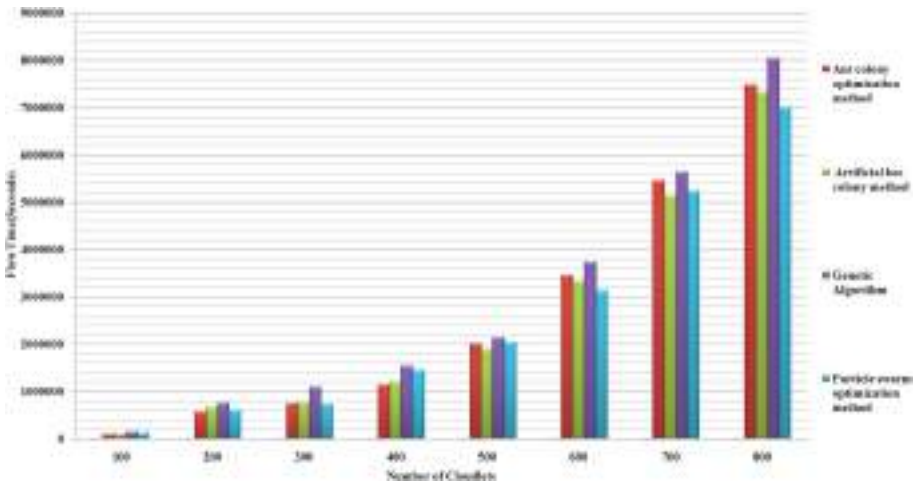

Fig. 10 Flow Time (Sec) outcomes for metaheuristic load balancing methods

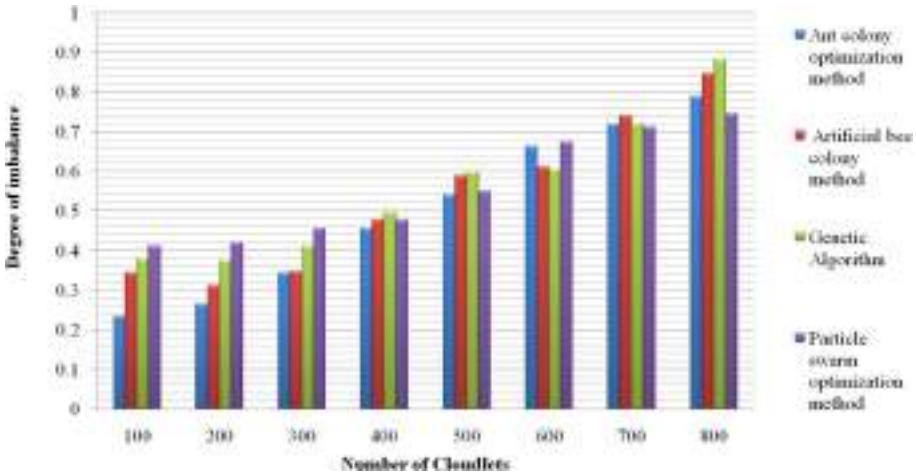

Fig. 11 Degree of Imbalance outcomes for metaheuristic load balancing methods

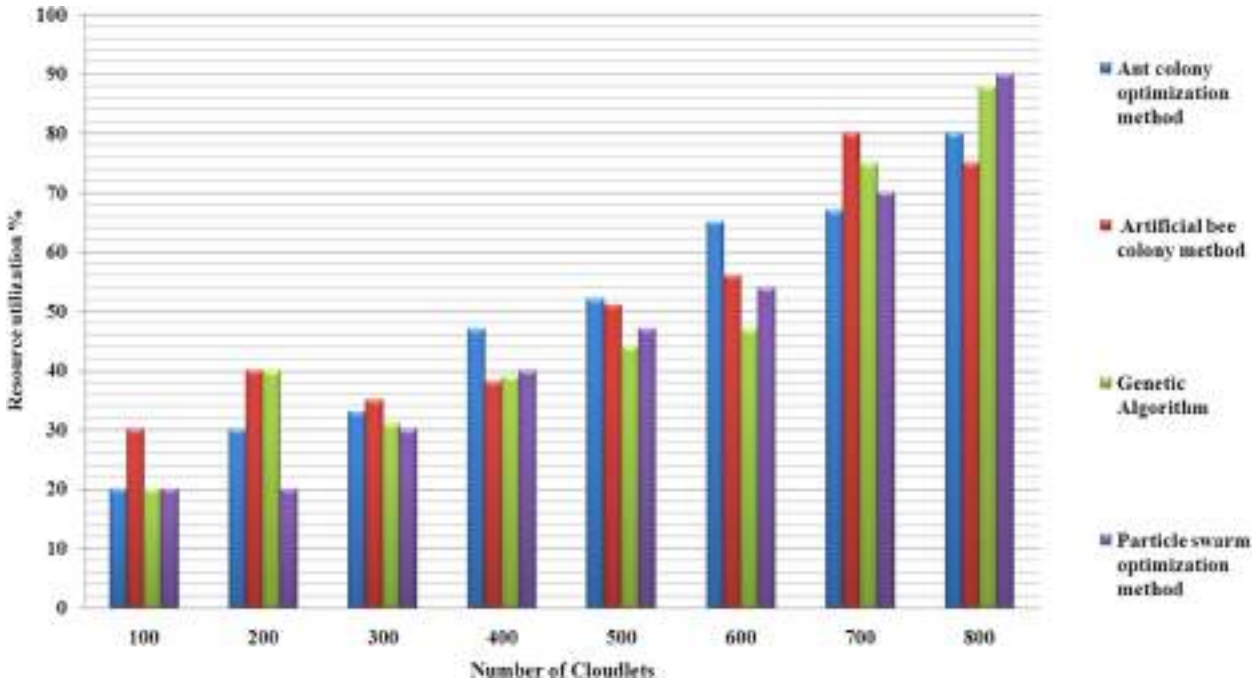

Fig. 12 Resource Utilization outcomes for metaheuristic load balancing methods

mean resource consumption. This article also discussed the taxonomy of cloud load balancing and metaheuristic methods, their challenges, issues, and applications. This article has also evaluated many important meta-heuristic methods for distributing resources in Cloud infrastructure. Various approaches to optimize the efficiency of meta-heuristics have been regarded. However, few meta-heuristic ways alone can accomplish the highest accuracy and efficiency of other optimization methods in cloud-based solutions' load balancing and resource allocation issues.

Numerous open issues that can be carried up for future investigation are also addressed. The simulation results are also calculated for various popular Metaheuristic load balancing methods, i.e., Ant colony optimization method, Artificial bee colony method, Genetic Algorithm, Particle swarm optimization method, and other performance measuring parameters, i.e.,

Makespan time, degree of imbalance, response time, data center processing time, flow time, and resource utilization. The Particle swarm optimization method performs better in improving makespan, flow time, throughput time, response time, and degree of imbalance. In future work, we will develop a more efficient method for cloud load balancing using existing Metaheuristic methods. The proposed approach will be compared with more load-balancing methods in real-time.

Abbreviations

CSP	Cloud service providers
CLB	Cloud load balancing
VMs	Virtual machines
DC	Data Center
CC	Cloud Computing
Quality of Services	QoS
CSU	Cloud Service Users
OLB	Opportunistic Load balancing
DCN	Data center network
CSA	Cuckoo Search Algorithm
IWO	Invasive Weed Optimization
AIS	Artificial Immune System
SHO	Spotted Hyena Optimization
DE	Differential Evaluation
ACO	Ant colony optimization
GP	Genetic Programming
EP	Evolunatory Programming
ABC	Artificial Bee Colony
SI	Swarm Intelligence
PSO	Particle Swarm Optimization
NIST	National Institute of Standards and Technology

Acknowledgements
We would like to thank all who directly and indirectly support this research.

Authors' contributions
Conceptualization by Jincheng Zhou, Tao Hai, Dayang Norhayati Abang Jawawi and Umesh Kumar Lilhore; Methodology by Jincheng Zhou, Tao Hai and Mounir Hamdi; Software by Poongodi M and Umesh Kumar Lilhore; formal analysis by Poongodi M, Mounir Hamdi and Cresantus Biamba; Investigation by Umesh Kumar Lilhore and Sarita Simaiya; Resources and data collection by Deema mohammed, Sarita Simaiya; Writing by: all the authors; Validation by: all the authors. All authors read and approved the final manuscript.

Funding
No funding received.

Availability of data and materials
The supporting data can be provided on request.

Declarations

Ethics approval and consent to participate
The ethics department approves the research of the School of Computer and Information, India.

Consent for publication
There search has consent from all authors, and there is no conflict.

Competing interests
The authors declare no competing interests.

Received: 19 August 2022 Accepted: 29 September 2022
Published online: 13 October 2023

References

1. Thakur A, Goraya MS (2022) RAFL: a hybrid Metaheuristic based resource allocation framework for load balancing in the cloud computing environment. Simul Model Pract Theory 116:102485
2. Sefati S, Mousavinasab M, ZarehFarkhady R (2022) Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation. J Supercomput 78(1):18–42
3. Singh RM, Awasthi LK, Sikka G (2022) Towards metaheuristic scheduling techniques in cloud and fog: an extensive taxonomic review. ACM Computing Surveys (CSUR) 55(3):1–43
4. Gopu A, Venkataraman N (2021) Virtual machine placement using multi-objective bat algorithm with decomposition in the distributed cloud: MOBA/D for VMP. Int J Appl Metaheuristic Comput 12(4):62–77
5. Swarnakar S, Bhattacharya S, Banerjee C (2021) A bio-inspired and heuristic-based hybrid algorithm for effective performance with load balancing in cloud environment. Int J Cloud Appl Comput 11(4):59–79
6. Biswal B, Shetty S, Rogers T (2015) Enhanced learning classifier to locate data in cloud data centres. Int J Metaheuristics 4(2):141
7. Singh H, Tyagi S, Kumar P, Gill SS, Buyya R (2021) Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: analysis, performance evaluation, and future directions. Simul Model Pract Theory 111(102353):102353
8. Bothra SK, Singhal S (2021) Nature-inspired metaheuristic scheduling algorithms in the cloud: a systematic review. Sci tech j inf Technol mech opt 21(4):463–472
9. Kumar J, Singh AK (2021) Performance evaluation of metaheuristics algorithms for workload prediction in cloud environment. Appl Soft Comput 113:107895. https://doi.org/10.1016/j.asoc.2021.107895
10. Gokalp O (2021) Performance evaluation of heuristic and metaheuristic algorithms for independent and static task scheduling in cloud computing. 2021 29th Signal Processing and Communications Applications Conference (SIU)
11. Ma L, Xu C, Ma H, Li Y, Wang J, Sun J (2021) Effective metaheuristic algorithms for bag-of-tasks scheduling problems under budget constraints on hybrid clouds. J Circuits Syst Comput 30(05):2150091
12. Sarma SK (2021) Metaheuristic based auto-scaling for microservices in cloud environment: a new container-aware application scheduling. Int J Pervasive Comput Commun. Ahead-of-print, no. ahead-of-print
13. Ramathilagam A, Vijayalakshmi K (2021) Workflow Scheduling in cloud environment using a novel metaheuristic optimization algorithm. Int J Commun. Syst 34(5):e4746
14. Zhang T, Lei Y, Zhang Q, Zou S, Huang J, Li F (2021) Fine-grained load balancing with traffic-aware rerouting in datacenter networks. J Cloud Comput Adv Syst Appl 10(1):1–20
15. Nandal P, Bura D, Singh M, Kumar S (2021) Analysis of different load balancing algorithms in cloud computing. Int J Cloud Appl Comput 11(4):100–112
16. Saxena D, Singh AK, Buyya R (2022) OP-MLB: an online VM prediction-based multi-objective load balancing framework for resource management at cloud data center. IEEE Trans Cloud Computing 10(4):2804–2816. https://doi.org/10.1109/TCC.2021.3059096
17. Malviya DK, Lilhore UK (2018) Survey on security threats in cloud computing. Int J Trend Sci Res Dev 3(1):1222–1226
18. Lilhore UK, Simaiya S, Guleria K, Prasad D (2020) An efficient load balancing method by using machine learning-based VM distribution and dynamic resource mapping. J Comput Theor Nanosci 17(6):2545–2551
19. Liu Z, Zhao A, Liang M (2021) A port-based forwarding load-balancing scheduling approach for cloud datacenter networks. J Cloud Comput Adv Syst. Appl 10(1):1–4
20. Lilhore UK, Simaiya S, Maheshwari S, Manhar A, Kumar S (2020) Cloud performance evaluation: hybrid load balancing model based on modified particle swarm optimization and improved metaheuristic firefly algorithms. Int J Adv Sci Technol 29(5):12315–12331
21. Hu Y, Wang H, Ma W (2020) Intelligent cloud workflow management and scheduling method for big data applications. J Cloud Comput Adv Syst Appl 9(1):1–3
22. Xuan Phi N, Ngoc Hieu L, Cong Hung T (2020) Load balancing algorithm on cloud computing for optimizing response time. Int J Cloud Comput Serv Archit 10(3):15–29

23. Diallo M, Quintero A, Pierre S (2021) An efficient approach based on ant colony optimization and Tabu search for a resource embedding across multiple cloud providers. IEEE Trans cloud computing 9(3):896–909
24. Lilhore U, Kumar S (2016) Modified fuzzy logic and advance particle swarm optimization model for cloud computing. Int J Mod Trends Eng Res (IJMTER) 3(8):230–235
25. Hu C, Deng Y, Min G, Huang P, Qin X (2021) QoS promotion in energy-efficient datacenters through peak load scheduling. IEEE Trans Cloud Comput 9(2):777–792
26. Sun H, Wang S, Zhou F, Yin L, Liu M (2023) Dynamic deployment and scheduling strategy for dual-service pooling-based hierarchical cloud service system in intelligent buildings. IEEE Trans Cloud Comput 11(1):139–155. https://doi.org/10.1109/TCC.2021.3078795
27. Liu C, Li K, Li K (2021) A game approach to multi-servers load balancing with load-dependent server availability consideration. IEEE Trans Cloud Comput 9(1):1–13
28. Wei X, Wang Y (2023) Popularity-based data placement with load balancing in edge computing. IEEE Trans Cloud Comput 11(1):397–411. https://doi.org/10.1109/TCC.2021.3096467
29. Sinha G, Sinha D (2020) Enhanced weighted round-robin algorithm to balance the load for effective resource utilization in cloud environment. EAI Endorsed Trans Cloud Syst 6(18):166284
30. Le Ngoc H, ThiHuyen TN, Nguyen XP, Tran CH (2020) MCCVA: A new approach using SVM and kmeans for load balancing on cloud. Int J Cloud Comput Serv Archit 10(3):1–14
31. Shen H, Chen L (2020) A resource usage intensity aware load balancing method for virtual machine migration in cloud data centers. IEEE Trans Cloud Comput 8(1):17–31
32. Yu L, Chen L, Cai Z, Shen H, Liang Y, Pan Y (2020) Stochastic load balancing for virtual resource management in data centers. IEEE Trans Cloud Comput 8(2):459–472
33. Pawar N, Lilhore UK, Agrawal N (2017) A hybrid ACHBDF load balancing method for optimum resource utilization in cloud computing. Int J Sci Res Comput Sci Eng Inform Technol (IJSRCSEIT), ISSN: 2456 3307:367–373
34. Jankee C, Verel S, Derbel B, Fonlupt C (2016) A fitness cloud model for adaptive metaheuristic selection methods. In Parallel Problem Solving from Nature – PPSN XIV. Springer International Publishing, Cham, pp 80–90
35. Nesmachnow S (2014) An overview of metaheuristics: accurate and efficient methods for optimization. Int J Metaheuristics 3(4):320
36. Meng Z, Li G, Wang X, Sait SM, Yıldız AR (2021) A comparative study of metaheuristic algorithms for reliability-based design optimization problems. Arch Comput Methods Eng 28(3):1853–1869
37. Malathi V, Kavitha V (2022) Energy-aware load balancing algorithm for upgraded effectiveness in green cloud computing. In Expert Clouds and Applications. Springer Singapore, Singapore, pp 247–26
38. Pai M, Rajarajeswari S, Akarsha DP, Ashwini SD (2022) Analytical study on load balancing algorithms in cloud computing. In Expert Clouds and Applications. Springer Singapore, Singapore, pp 631–646
39. Sonekar SV, Kokate R, Titre M, Bhoyar A, Haque M, Patil S (2022) Load balancing approach and the diminishing impact of a malicious node in ad hoc networks. In Advanced Computing and Intelligent Technologies. Springer Singapore, Singapore, pp 523–536
40. Shukla S, Suryavanshi R, Yadav D (2022) Formal modelling of cluster-coordinator-based load balancing protocol using event-B. In Proceedings of Second Doctoral Symposium on Computational Intelligence. Springer Singapore, Singapore, pp 593–603
41. Ahmad S, Jamil F, Ali A, Khan E, Ibrahim M, KeunWhangbo T (2022) Effectively handling network congestion and load balancing in software-defined networking. Comput mater contin 70(1):1363–1379
42. Lilhore U, Kumar S (2016) Advance anticipatory performance improvement model, for cloud computing. Int J Recent Trends Eng Res (IJRTER) 2(08):210–215
43. Upadhyay R, Lilhore U (2016) Review of various load distribution methods for cloud computing, to improve cloud performance. Int J Comput Sci Eng 4:61–64
44. Khan T, Singh K, Hasan MH, Ahmad K, Reddy GT, Mohan S, Ahmadian A (2021) ETERS: a comprehensive energy aware trust-based efficient routing scheme for adversarial WSNs. Futur Gener Comput Syst 125:921–943. https://doi.org/10.1016/j.future.2021.06.049
45. PalanivelRajan D, Premalatha J, Velliangiri S, Karthikeyan P (2022) Blockchain enabled joint trust (MF-WWO-WO) algorithm for clustered-based energy efficient routing protocol in wireless sensor network. Trans Emerg Telecommun Technol. https://doi.org/10.1002/ett.4502,33,7

Publisher's Note