ISSN 2319 - 6629
Volume 12, No.5, August 2023 – September 2023
International Journal of Wireless Communications and Networking Technologies
Available Online at http://warse.org/IJWCNT/static/pdf/file/ijwcnt021252023.pdf
https://doi.org/10.30534/ijwcnt/2023/021252023

Review of Load Balancing Algorithms Inspired by Artificial Bee Colony Algorithm in the Cloud Computing

Hind Salem Alatawi[1], Sanaa Abdullah Sharaf [2]
[1]Computer Science Department, Applied college, Tabuk University, Tabuk, Saudi Arabia, hs.alatawi@ut.edu.sa
[2]Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia, sshara@kau.edu.sa

Received Date: July 28, 2023 Accepted Date: August 22, 2023 Published Date : September 07, 2023

ABSTRACT

Research interest in cloud-based load balancing and task scheduling has grown rapidly over the last few years. To be more precise, load balancing of virtual machine tasks (VMs) that require nature-inspired algorithms has become an area of particular interest. It is essential to ensure load balancing between VMs as this avoids overloading and underloading VMs, aspects that can cause issues like high-power consumption, increased execution time, and elevated response times, all of which can ultimately cause a system failure.
Swarm intelligence is critical when dealing with issues that are difficult to solve and must be overcome using traditional and mathematical techniques. The algorithm was developed by Karaboga in 2005 and is inspired by the foraging behaviors of an artificial bee colony. This algorithm is extremely robust, convergent, and flexible. Another researcher applied the ABC method to load balancing in order to enhance it.
In-depth research on load balancing in cloud computing utilizing the ABC method is presented in this review paper. Moreover, this work also discusses some fundamental ideas on the intelligence and characteristics of the swarm. Additionally, the paper provided a detailed explanation of cloud computing, its services, and components, as well as the concept and objectives of load balancing in the cloud.

Key words : Cloud Computing, Load Balancing, Honeybee approach, , Task Scheduling .

1. INTRODUCTION

Across the Information Technology (IT) industry, Cloud Computing (CC) has been regarded as a positive disruptive technology. Many companies have adopted CC due to its ability to lower costs and its dynamic allocation of resources [1]. However, although CC provides software, infrastructure, and platform access to consumers as a service, various problems such as performance unpredictability, resource rationing, unavailability of resources to meet all requirements, storage capacity, and security and data confidentiality issues remain [2]. Load balancing is thus a crucial concern to minimize response time and ensure reliability, as well as maximizing throughput and minimizing costs in the cloud environment [2]. Load balancing techniques are employed after task scheduling to achieve high performance and more efficient resource utilization [3]. Load balancing techniques distribute the load uniformly over the network interfaces, storage devices such as hard drives, servers, and other resource types within the cloud data center [4] . Load balancing and scheduling issues are NP-hard problems and require suitable algorithms to be developed to ensure optimal performance [5] .
Thus, the researcher created an ABC algorithm that could be used for scheduling. This algorithm enhances makespan and network stability results. Swarm intelligence (SI) refers to collective natural behavior in the form of decentralized, self-organized systems, or artificial systems. Furthermore, SI systems consist of simple agents that interact locally with each other and with their environment [6]. Usually, the inspiration is derived from nature, particularly biological systems. The agents follow very simple rules and there is no centralized control structure to determine how individual agents should behave. Thus, both local and random, interactions between such agents can cause “intelligent” global behavior to emerge that is unknown to the individual agents. Ant colonies, flocks of birds, herds of animals, schools of fish, and bacterial growth are all natural examples of SI [6] . SI can be used in many different fields, including research, social sciences, and engineering. Moreover, their ABC algorithms can be used in cloud computing to reduce load balancing. The key objectives of the present review are as follows:
1. To comprehensively assess load balancing algorithms that are based on the ABC approach and that are compatible with CC.
2. To determine the advantages and disadvantages of current methods.
3. To enable researchers to evaluate vital concepts in the field of load balancing algorithms that have not previously been examined.
This review paper examined the fundamental concepts on which this research is based. It first presented the concept of

CC and components, before discussing types of CC and the services these provide. It then explored the concept of load balancing, including its main types, and objectives, discussing the idea of a load balancing model and the various load balancing QoS metrics. The concept of the honeybee optimization algorithm and the concept of the ABC algorithm, along with its fundamental algorithmic structure, and a definition of meta-heuristics and swarm intelligence algorithms were offered; a bee colony's relationship with the cloud environment was also outlined.

2. CLOUD COMPUTING

2.1 Cloud Computing Concepts

Be CC can be used to provide various services, including utilities such as electricity, telecommunications, and water [7]. Different computing paradigms, including grid computing, mainframes, and clusters, have previously been used to provide adequate computing power, but the extensive use of distributed computing now requires many organizations to effectively store and retrieve substantial amounts of data. CC was thus developed as a computing platform that could simultaneously provide services to customers from many different domains [8] .

This model is called CC, a new form of ubiquitous computing developed from the concept of on-request access to online, shared collections of assets in a self- available, calculated, and increasingly versatile manner [9]. This model enables users to make use of cloud services as needed on a "pay-per-use" basis regardless of their location. Various data centers support this technology, and their virtualization advances have allowed them to ensure consolidation as well as significant use of resources [5]. In addition, utilizing a

CC paradigm can help consumers to subscribe to the most appropriate services by signing a contract with the cloud vendor, known as Service level Agreement (SLA),that outlines the relevant QoS indicators as well as the expected parameters for the services rendered [9] .

Buyya et al. described CC as being formed of parallel, distributed systems that include various virtualized, interconnected computers that are effectively provisioned to act as unified computing resources to match the level of service required. The National Institute of Standards and Technology (NIST) instead focused on the goals of CC by defining it as a model that provides on-demand and convenient net- work access to configurable computing resources such as networks, storage, servers, services, and applications, which are shared yet can be provided and released at a fast pace, requiring minimal service provider interaction or management effort [10]. Other cloud models not only focus on this availability but also involve five fundamental features, three service models, and four deployment models [8] . Figure 1 presents the NIST CC model. The four deployment models are thus public cloud, hybrid, community cloud, and private cloud, as discussed below:

(a) Private cloud: When resources are used through virtualization and resource management tools within a business's own premises, this may create a "private cloud." [11]. An example of a private cloud is the Eucalyptus system, whose major benefit is that it offers easy security management, upgrades, and maintenance as well as more control over how it is deployed and used [12].

(b) Public cloud: Access to this type of cloud is achieved over the Internet. In public clouds, applications are either developed in the cloud or transferred from other infrastructures to obtain CC benefits. Microsoft Azure and Google App Engine offer examples of public clouds [12] .

(c) Community cloud: The infrastructure and services in a community cloud are shared and accessed by organizations with similar interests [13].

(d) Hybrid cloud: Public and private cloud models can be combined in hybrid clouds. A hybrid cloud environment can thus offer an on-demand and externally provisioned scale of services . Improving a private cloud by using a public cloud resource can be particularly helpful in addressing unexpected workload surges. Amazon Web Services is an example of a hybrid cloud [13] .

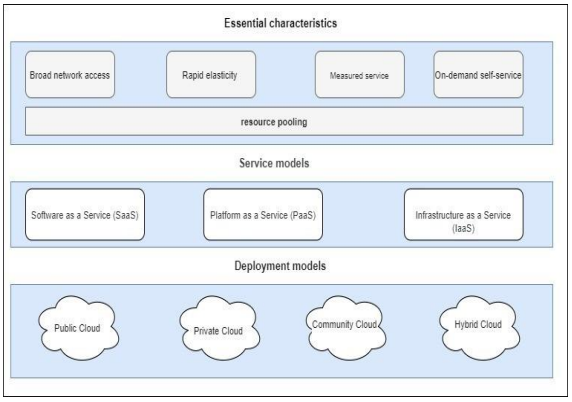

Figure 1: NIST Model for Cloud Computing [5**]**

2.2 Cloud Components

There are three main components in the cloud: clients, distributed servers, and data centers. Figure 2 illustrates the components mentioned that form a CC solution [14]. Every element serves a particular purpose and plays a particular role.

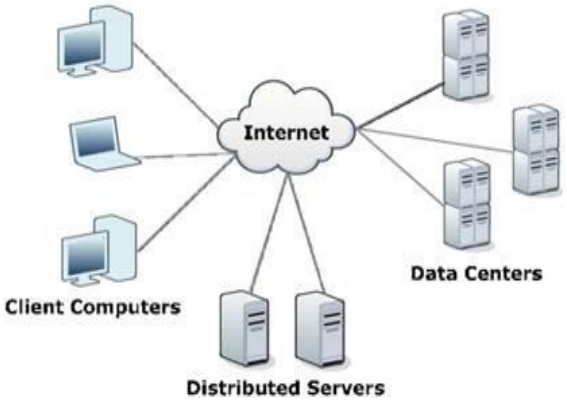

Figure 2: Components of Cloud Computing [14]

1. Clients

Clients refers to the devices with which end users interact to manage their information within the cloud. There are three major types of clients [7] :

• Mobile: These include PDAs and smartphones, such as Blackberries, iPhones, or Windows Mobile Smartphones.

• Thin Clients: This refer to computers with no internal hard drives that access the server and require it to do all the work; the thin client then simply displays the relevant information.

• Thick Clients: These are regular computers that connect to the cloud using a web browser such as Internet Explorer or Firefox. It should be noted that thin clients have become increasingly popular due to their reduced environmental effects [8] .

2. Datacenter

The data center is the label for a set of servers that host the various applications . Modern virtualization concepts can also be implemented for installing any software that requires more than virtual server application instances [15].

3. Distributed servers

A server, which actively checks the services of their hosts, known as distributed server. Distributed servers are the part of a cloud which is available throughout the internet hosting different applications. But while using the application from the cloud, the user would feel that he /she is using this application from its own machine [16].

2.3 Cloud Computing Services

A CC is a system that is formed of three services: Software As a Service (SaaS), Platform As a Service (PaaS), and Infrastructure As a Service (IaaS) [17]. The service types offered to the four cloud structures are illustrated in figure 2.3 . The three categories of service are further defined below:

IaaS: This provides computer hardware to an organization as a service that can dis- tribute and run the required software, including operating systems and applications such as firewalls [17]. Further, IaaS utilizes virtualization technology to transform physical resources into logical resources that can be provisioned dynamically and accessed by customers based on their requirements. Google, Verizon, Rackspace Cloud Servers, Amazon Elastic Compute Cloud, and IBM are among the major companies that offer infrastructure as a service [18].

PaaS : This is an advanced form of CC service that offers a layer of software or an enclosed development environment that can be used as a basis for developing higher levels of service. Some well-known examples of PaaS include LAMP platforms (Linux, Apache, MySQL, and PHP), restricted J2EE, and Ruby [17]. Google's App Engine and Force.com offer popular PaaS examples. PaaS services involve design, hosting applications, development, collaboration, security, database integration, scaling, and web service integration. When using such services, users therefore do not have to be concerned about owning the correct software and hardware resources or hiring experts to manage those resources. Such schemes offer flexible software installation along with scalability [18].

SaaS : This is a software distribution model that allows users access via internet hosting. Providers thus not only develop all infrastructures, hardware, software, and operating systems, but also provide services such as post- maintenance [18]. Customers generally perceive SaaS models as web-based application interfaces that use the internet to provide services that are accessed through a web browser. A wide range of devices such as laptops and smartphones can be used to access hosted applications on such services, including Gmail and Google Docs. Unlike traditional software, in SaaS, the customer is not required to purchase licenses or to install, maintain, run, or upgrade software on their own computer [17].

3. LOAD BALANCING IN THE CLOUD

3.1 Load Balancing Overview

In a cloud-based environment in which requests concerning platforms and services are received at different times, the load on the servers must be balanced [19] . The various load types include CPU load (the sum of those processes presently being run and those waiting to be run), memory use, and network delay load (which is the time required for data to travel from one node to the next across the network) [19] .

Load balancing—also known as traffic management in the CC context refers to managing total load across individual nodes in order to increase the well-being of the collective system, thereby enhancing resource utilization [19] .

At present, most organizations are leaning toward using cloud-side services, which has increased the number of cloud users. Load balancing is thus increasingly necessary for maintaining the load on the server [16] . Load balancing ensures that the workload is distributed among various nodes so that capacity is maintained and there is no overloading or underloading on any given node [12] . There are two major tasks that load balancing must thus focus on: providing resources and scheduling tasks in a disseminated environment, and in busy CC environments, such load balancing is a significant concern [7] . Figure 3 offers a basic illustration of load balancing:

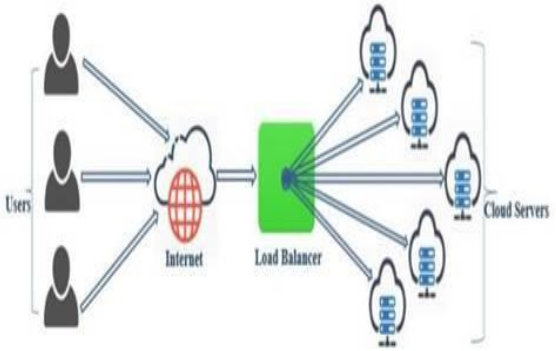

Figure 3: Load Balancing Process [13]

As shown in figure 3, users must first request access to the cloud server. Each request is then sent over the internet to the load balancer, after which the load balancer chooses where the demand will be executed by identifying an appropriate VM. The load balancer computes the load on each VM to ensure no VM is underloaded or overloaded [20] .

3.2 Aims of Load Balancing

Through load balancing, workloads can be distributed efficiently in terms of the resources available [20]. This also allows for a continuation of service if any of the service's components fail, as load balancing can provision and de-provision application instances as well as improving resource utilization [11]. Further, load balancing can help reduce task response time and enhance the use of resources, thus improving system performance while keeping costs low [17] . It can also ensure flexibility and scalability for applications that may increase in size in the future and thus require more resources while simultaneously prioritizing those jobs that must be executed instantly [17]. Moreover, load balancing can decrease energy consumption and carbon emissions, provide access to more resources, avoid bottlenecks, and fulfill QoS requirements [19]. It is, however, necessary to develop proper workload mapping as well as load balancing techniques that take various metrics into consideration.

3.3 Load Balancing Process in CC

Within the cloud, on-demand access can be provided to shared resources, such as networks, servers, and storage, which requires both the user's resources and workload to be managed and controlled [21]. To manage the available resources in light of user requests, it is important to utilize an efficient load balancer that can assign tasks to appropriate VMs according to the relevant QoS requirements [18] .

The load balancing process in CC is shown in figure 4, where we can see the load balancer receives users' requests and runs load balancing algorithms to distribute the requests among the VMs [19]. The load balancer decides which VM should be assigned to the next request [22]. The data center controller (DCC) is in charge of task management. Tasks are submitted to the load balancer, which performs load balancing algorithm to assign tasks to a suitable VM [11]. A VMs process the requests of the users. Users are located all around the world and their requests are submitted randomly. A hypervisor or Virtual Machine Monitor (VMM) is used to create and manage the VMs. A VMM provides four operations: multiplexing, suspension (storage), provision (resume), and life migration [23] . These operations are necessary for load balancing.

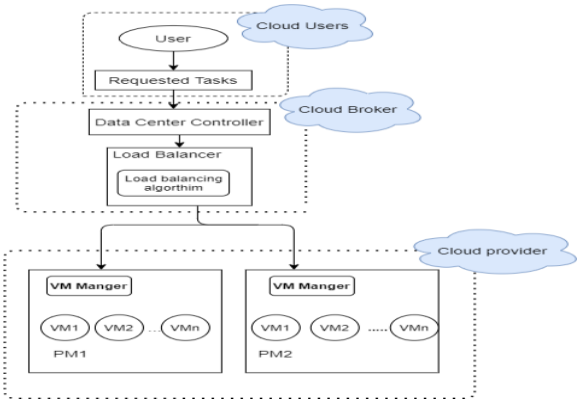

Figure 4: Load Balancing Process [19]

3.4 Load Balancing QoS Metrics

Load balancing can help improve qualitative metrics such as performance, scalability, resource utilization, fault tolerance, processing time, migration time, and response time in CC environments, thus improving QoS for customers [23]. It is thus necessary to identify the fundamental load balancing metrics re- quired to assess the different load balancing algorithms' performance . The main CC environment QoS performance metrics that impact load balancing are:

- Throughput: The number of tasks and executions completed in a stipulated period of time. High throughput ensures better performance [3] .
- Response Time: It is the time required by the system to respond a task. In other words, it is the amount of time taken between the submission of a request and the first response which is produced [8].
- Makespan: This refers to the overall time needed to complete all tasks sent into the system. The system's makespan is thus also the time that data center functions take to perform [16] .
- Fault tolerance: This is a system characteristic that allows work to continue despite component failure. Knowing the number of failure points, and whether they are single point failures or multipoint failures, can help assess the extent of fault-tolerance [24]. Service providers need more resources or VMs to overcome certain cloud system limitations; this can, however, result in extra costs in the search for a fault-free system [9].
- Migration time: This is the time taken for a task or a VM to be migrated from one resource to a second resource. Tasks may be migrated from one VM to another within a single host or across different hosts [5]. Further, VMs may be migrated from one host to a different host in the same or different data centers [13]. In cases where tasks need resources from different VMs or where there are interruptions in task execution, tasks may be migrated. Similarly, if a VM crashes during execution, that VM will be migrated to a different host [23]. Additional VM migrations leads to higher migration time, thus degrading the system's makespan and de-optimizing load balancing [17] .
- Degree of Imbalance: This assesses the VMs' imbalance of work. For example, a schedule is said to best if it is close to 0 degrees of imbalance [4] .
- Reliability: This refers to tasks being transferred to another VM if there is system failure to enhance system reliability [4]. Increased reliability also improves system stability [25].
- Resource utilization: This checks machine usage of all resources; this utiliza- tion should be high for greatest efficiency [22].
- Scalability: This is a system or model feature that may describe the capacity for responding to unpredictable situations [22]. It refers to the extent to which a balanced system can survive in case of increased amounts or sizes of tasks or workloads. Resources are regularly rescaled in scalable cloud systems to promote efficiency [11] .

3.5 Types of Load Balancing

Based on the common system conditions, there are two basic types of load balancing algorithms, which will be discussed in detail in this section.

1. Static load balancing techniques

Static techniques such as Weighted Round Robin (WRR) and threshold algorithm, tend to involve a degree of previous knowledge as well as certain assumptions regarding the system's global status, which may include job resource requirements, system nodes' processing power, communication time, memory, and the capacity of storage devices [13].

A major disadvantage of static load balancing algorithms is that the system's current state is not taken into account in any reassignment decisions, and it thus cannot be regarded as an appropriate approach for systems where the majority of states are subject to dynamic change [26].

2. Dynamic load balancing techniques

Dynamic load balancing techniques, including the Honey Bee Foraging Algorithm and ACO Algorithm are applied based on the existing state of the system, and thus no prior knowledge is needed. Such techniques involve tasks being dynamically moved from overloaded to underloaded nodes [24]. Thus, dynamic load balancing algorithms offer the benefit of constant positive change based on the system's existing state. Dynamic mechanisms can thus provide better performance with more efficiency and accuracy; however, it is significantly more difficult and complicated to design and apply a dynamic load balancing algorithm than to determine a static solution [19]. There are two main dynamic load balancing algorithm types: distributed and non-distributed. Load balancing in distributed approaches can be performed by each of the system's nodes, and nodes can interact to achieve load balancing. In terms of non-distributed balancing, each single node or a group of nodes executes a specific load balancing task [11].

3.6 Meta-heuristic Algorithms

A major difficulty with load balancing in a cloud environment is ensuring the proper load is provided to each VM to avoid any VM becoming underloaded or overloaded. Moreover, it is important to map the tasks to those VMs that can handle them [18]. One effective method for addressing complicated problems such as load balancing is the application of meta-heuristic algorithms. Meta-heuristic algorithms tend to be dynamic, and they include evolutionary as well as swarm intelligence algorithms [23].

Meta-heuristics originated in the fields of Artificial Intelligence (AI) and Operations Research. Using heuristic techniques does not deliver near-optimal solutions, as such techniques only provide a limited number of solutions [13]. A major drawback with heuristic methods is the focus on poor quality local optima; this triggered the development of meta-heuristics as an iterative improvement [13] .

Figure 2.6 presents the classification of the various meta-heuristic algorithms used in load balancing. These algorithms are local Search, single solution, evolutionary algorithms, swarm-based algorithms, and population-based algorithm [18] .

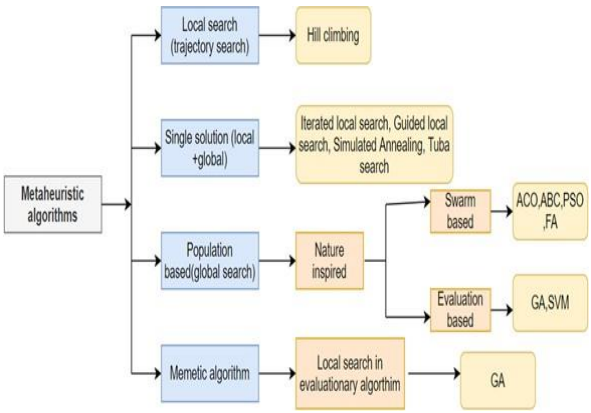

Figure 5: Meta-heuristic Algorithms Classification [10]

3.7 Meta-heuristic Algorithms

SI is an innovative method of problem solving based on the collective intelligence of swarms of biological populations as well as the social-behavioral model of insects and similar animals [13]. SI refers to any behavioral computational paradigm that can address distributed problems by managing the interactions of simple information-processing units [16].

SI is rapidly gaining popularity among engineers, computer scientists, bioinformaticians, economists, and operational researchers due to the fact that there are significant counterparts to the problems solved by natural intelligent swarms (such as finding food, building nests, and dividing labor among nestmates) in various different engineering areas in the human world [23]. Such techniques can thus help solve numerous combinatorial optimization and search problems, and since the early 2000s, researchers have taken an interest in the behavior of swarm systems as models for developing new intelligent approaches [23].

In the SI field, the most popular algorithms include ACO, ABC, and particle swarm optimization [17]. Features such as division of labor and self- organization, along with the satisfaction principles necessary in SI, are particularly evident in honey bee colonies [17]. The following section will thus discuss the ABC algorithm in more detail.

3.8 Artificial Bee Colony Optimization

The ABC is based on honey bees' intelligent foraging behavior as they seek food sources; Dervis first presented this method for addressing real-world problems in 2005 [27]. ABC refers to a subsection of the swarm-intelligence based algorithms that address different optimization problems by imitating the honeybee swarms' collective intelligence [27]. A bee gathers food from a specific flower, or food source, and a colony of bees develops where such bees cooperates to find better food sources. Sharing information helps with making decisions and examining the search space [28].

The information exchange that takes place among the honey bees is crucial to the success of the foraging behavior that is the most significant feature of a honey bee [7]. Bees leave the hive to search for food sources; after finding a food source, the bee extracts a quantity of nectar and stores it in its stomach

before returning to the hive [28]. Next, the bee shares information about the source of the nectar with the other bees by performing a dance, which informs the other bees about the food in terms of location, amount of nectar, direction, and food quality [10] .

Several different types of dances are thus performed in a hive, and the other bees can also touch the dancing bee with their antenna to taste the nectar [11]. There are three main types of dances that are used, however. The first is the round dance that is used when the food source is close to the hive, in which the distance to the food source is not conveyed [5]. The second is the waggle dance, which is used when the food source is far from the hive and which informs the other bees about both the food source and its direction [16] . The third type is the tremble dance, which is performed when there is likely to be a delay with the food source, allowing other bees to be informed both of its present position and of the need for delay [13].

3.9 Algorithmic Structure of ABC

The ABC algorithm has been applied in various fields, including job shop schedul- ing, binary optimization, load balancing, as well as traveling salesman digital signal processing [20]. There are three groups of bees in an ABC system, namely employed bees, scouts, and onlookers; each group performs different functions.

1. Employed bees: The employed bees are also called leader bees [10] . They seek food sources and come back to their hives and perform a dance to convey information regarding the foraged food sources, including the distance, quantity, and direction, as well as quality of the food source [20] .
2. Scout bees: These bees randomly search for new food sources around the hive [27]. Scout bees conduct random searches to find new food sources near the hive, and whenever they come across an existing food source, they begin a new search for a new source in that environment [27] .
3. Onlooker bees: Onlooker bees gather information from the employed bees in the hive and select food sources based on these dances [13]. Both scouts and onlookers are also known as unemployed bees [16].

The ABC optimization approach's basic algorithmic structure is thus [15]

1: Procedure Initialization Phase
2: while Cycle≠Maximum Cycle Number or ≠ Maximum CPU time do
3: Employed Bees Phase
4: Onlooker Bees Phase
5: Scout Bees Phase

In the initialization phase, artificial scout bees initiate the search for food sources (solutions) [29].

In the employed bee phase, artificial employed bees seek new food sources with more nectar near the previously memorized food sources [29]. Upon identifying a neighboring food source, its fitness is assessed, and if the new food source is confirmed, greedy selection is implemented between it and the original source [28]. Next, the employed bees provide food source information to the onlooker bees who are waiting in the hive [29].

In the onlooker bee phase, artificial onlooker bees select their food sources proba- bilistically based on the employed bee information. This can include a fitness-based selection technique [29]. Once a food source is probabilistically selected for an onlooker bee, a neighborhood source is identified and its fitness value assessed. As in the employed bee phase, a greedy selection is implemented to make a choice between the two sources [29] .

In the scout bee phase, employed bees who provide solutions that cannot be enhanced abandon their solutions and turn into scouts [30]. These new scouts begin randomly seeking new solutions, ensuring that sources that were initially poor or have been made poor through exploitation are abandoned over time [30] .

3.10 Bee Colony Based Load Balancing Algorithm

The load balancing of tasks can be implemented by adopting the concepts underlying honeybees' foraging behaviors [14] . CC systems and a colony of honey bees foraging for and harvesting food share several similarities [31] a: cloud computing's system target is similar to that of a bee colony in that CC seeks maximum system throughput while honeybees require as much nectar as possible [30].

Tasks are equivalent to individual honeybees, while the VMs can be regarded as food sources [22]. A VM's task loading can thus be compared to the way in which honeybees forage various food sources, such as a patch of flowers [24]. In case of a VM becoming overloaded, which resembles a food source being depleted, the task must be scheduled to a VM that is underloaded, in the same way a foraging honeybee must identify new sources of food [24] . The task that is removed can also update other tasks regarding the original VM's overloaded status, in the same way honeybees communicate with bees in their beehive [32] . The task provides updates about VM status in terms of the number of tasks that the VM is processing as well as the number and details of high priority tasks that the VM is presently processing [24] , and such updates provide effective information for determining which tasks must be allocated to which VMs according to VMs' availability and loading, just as honeybees decide which food sources to visit according to the information about availability offered by their fellow bees [24]. The relationships between a bee colony and a cloud environment are outlined in Table 1 [32] .

Table 1: The Relationship Between a Bee Colony and a Cloud Environment

Honeybee Hive	Cloud Environment
Honeybee	Task (Cloudlet) (cloud users)
Food source	VM or Host Machine (cloud provider)
Bee foraging a food source	Loading of a task to a VM
Bee finds depleted food source	VM is overloaded
Foraging bee finds a new food source	Removed task is scheduled to an under loaded VM (cloud scheduler or cloud broker)

4. LOAD BALANCING IN CLOUD COMPUTING USING ABC TECHNIQUE

In this section we present a review of references that have provided solutions to the problem of load balancing in the cloud using the ABC approach. We divided the reviewed research into two sections: The first section focused on improving service quality factors in the load balancing algorithm and the other section focused on the improvement the load balancing process by reducing energy consumption.
Table 2 displays a comparison of the reviewed studies in terms of approach, disadvantage, advantage, environment, and decision parameter.

4.1 Honeybee Behavior Algorithms Based on QoS Factors

In [30], an ABC-based load balancing mechanism developed through imitation of the behavior of honeybees optimizes the amount of nectar (i.e. system throughput) to reach the maximum throughput. The findings suggest that the approach can enhance scalability and throughput but has a high response time. In addition, it provides better stability for a certain number of requests compared to others.
An extra step has also been extended a mutation operator within an ABC in [31]. Once the employed bees assess the solution space, the mutation is implemented. The chosen food source is random, and the mutation operator is implemented if the mutation is satisfied. It is possible to alter the local best position using mutation, with optimal localization of the algorithm. The mutation operator is examined to develop new food sources. Thus, better fitness value concerning the newly generated algorithm leads to the replacement of the older ones. Results suggest improved execution time compared with genetic algorithm. On the other hand, only one of the QoS factors, execution time, is taken into account by this algorithm. The present study is concerned with the job scheduling problem. This problem is related to the allocation of the jobs in the system in such a way so as to achieve optimization of the general application performance and at the same time to ensure result validity. However, this problem overlooks the matter of load balancing.
L.D and Krishnab in [24] developed the HBB-LB algorithm, which enables load balancing and considers the priorities of tasks taken away from VMs with heavy loads. Task migration was used by the HBB-LB algorithm to keep VMs in balance. The tasks that are extracted from overloaded VMs act like honeybees. Once submitted to the underloaded VM, the different priority tasks as well as the tasks allocated to the VM are updated. This information can also benefit other tasks. For example, every time a high-priority task should be submitted to VMs, it is important to take into account the VM with the fewest high-priority tasks to ensure that the specific task can be completed sooner. The tasks that are extracted are submitted to underloaded VMs, as all VMs are arranged in ascending order. Comparing the proposed algorithm with WRR, first in first out, and dynamic load balancing indicated good results with no additional overhead. This technique effectively balances non-preemptive independent tasks and improves makespan and response time. Low-priority tasks are a constant occurrence, despite the high throughput. However, the authors did not examine power consumption. The HBB-LB algorithm's flaw is that the balancing mechanism begins only when the whole system is unbalanced, so the degree of imbalance in this algorithm is high. Nevertheless, despite establishing priority as the key QoS parameter, other QoS factors are disregarded in load balancing process by this algorithm.
In[33], with modifications to the capacity equation and load, Vasudevan et al. (2016) improved the HBB-LB algorithm.

The enhanced version of HBB-LB illustrates that both dependent and independent tasks have improved performance with reduced makespan. The HBB-LB algorithm can concentrate on the priorities of the task needing to be allocated to the VMs. Being able to use a priority notion means that the VM has a decreased response time along with improved throughput.
The HBB-LB algorithm takes into account only the load conditions in the selection of the VM and does not consider other crucial QoS factors needed to improve efficiency in cloud computing. Consequently, an Improved Honeybee Behavior based Load Balancing (IHBB-LB) algorithm was developed as illustrated in [11]. It included more QoS parameters of the VM, such as reliability and cost, to improve load balancing. The results illustrate that the IHBB- LB algorithm was more efficient than HBB-LB in terms of makespan and the number of migrated tasks. According to the findings, the degree of imbalance was 1.45 and 1.43 for HBB-LB and IHBB-LB, respectively, reflecting a minor improvement in the degree of imbalance associated with IHBB-LB. Since the IHBB-LB evaluates its output on a limited number of tasks (40), the value of degree imbalance is still considered a high value .
In [34] , Shobana et al. (2014) formulated a method for preemptive task scheduling in CC, involving the calculation of the VM load based on the decision parameters of processor and bandwidth. The conditions for preemption to happen is for the priority of tasks removed from VM with a heavy load to exceed that of the running tasks and for the removed tasks to have a lower anticipated completion time compared to the running tasks. When these conditions are fulfilled, preemption of the running tasks occurs and the preempted task state is stored in the process control block. The next step is the execution of the incoming task in a suitable VM, followed by the resumption of an interrupted task from the status at which it was halted. Task distribution to VM is accompanied by information update, including the number of allocated tasks and VM priority, as well as the overloaded VM set, underloaded VM set, and balanced VM set . The VM is included in the balanced set when its threshold value is reached. When every VM is migrated to a balanced VM set, load balancing is considered to have been a success. The priority of the task and its forecasts about decreasing latency and increasing efficiency is determined by the algorithm.

Through the efficient usage of resources, users' response times are enhanced. However, no comparison has been undertaken between the suggested algorithm and other algorithms. The number of task migrations and the degree of imbalance is not considered in this work. Further, when selecting a VM, this algorithm considers only the load conditions and overlooks other QoS factors.

In [35] , Sheeja and Jayalekshmi (2014) integrated the principles of honeybee behavior for load balancing and the Pareto dominance relationship for identification of the ideal VM from a series of underloaded VMs into one algorithm, namely, Honeybee Behavior based Dynamic Load Balancing with Pareto dominance (HBBLBP). In this algorithm, if there are multiple underloaded VMs for task processing, the tasks are allocated to the VM that fulfills the Pareto dominance principle, which specifies that an alternative should be selected based on whether it yields a better result compared to a different alternative. In this case, the selection of the ideal VM is based on a comparison of the task execution cost and anticipated task running time on different VMs. In this way, the method improves both execution time and cost of VM used, by contrast to FCFS and HBB-LB. Furthermore, in HBBLBP, the key factor of QoS is cost, while other factors are not considered. The algorithm also has a high degree of imbalance is 3.11 as well as a high number of migrated tasks (14 out of 30).

In [36] ,George et al. (2017) proposed a new algorithm, namely, the enhanced honeybee-inspired load balancing algorithm. In the HBB-LB algorithm, task distribution to VMs is based on the number of tasks currently running in the VM. By contrast, in the new algorithm, weights are assigned to VMs according to the latter's computing power, with the VM with the highest computing power being allocated a weight of '1', the VM with the second-highest computing power being allocated a weight of '2', and so on. Thus, tasks are allocated to VMs depending on the VM resource needs. Another aspect taken into account in the allocation process is task priority. The outcomes of experiments have indicated that, by comparison to HBB-LB, the enhanced honeybee-inspired load balancing algorithm provides a better overall response time as well as better data center processing time. However, the algorithm did not observe the degree of system imbalance, or the number of tasks migrated. This research, as well as previous studies, overlooks the power consumption factor.

LBA-HB has been proposed in [22] being based on the premise that a load balancing algorithm will efficiently allocate the dynamic workload across all the hosts in the cloud to enhance the utilization of resources and execution time efficiencies. Results from the simulation demonstrate that the LBA-HB algorithm reduced the average response and execution times compared to the more popular algorithms, namely, RR and Modified throttled. Furthermore, LBA-HB algorithm avoids task migration. The authors did not provide a clear explanation for why their algorithm does not allow task migration. The degree imbalance of LBA-HB is 1.58, but this has to be reduced further. The clear disadvantages in using the LBA-HB approach are:

- Choosing the appropriate host is only based on processing time, although other factors affect its performance, including load and capacity of the host.
- The VM used to accept the task is reliant on a single element (the number of tasks handled by VMs) even though other crucial factors should be thought about in the context of load balancing, such as power consumption and cost. Moreover, a VM containing a great number of tasks does not automatically imply consumption of large amounts of resources and power [28]. By including a number of QoS factors of hosts and VMs, the performance of the LBA-HB process can be improved.

In [29], Thanka et al., (2017) an Improved Efficient-ABC (IE-ABC) algorithm is presented for a QoS and security program in the cloud environment. The ABC algorithm is altered for efficient service and security-aware scheduling. A task is assigned to the best VM, depending on the user's QoS policies and critical level of security. The outcome is improvement in the execution time compared to ABC by 19.9%. Despite this, the power consumption factor is not considered and number of task migration is high with 14 out of 30 tasks being migrated.

A job scheduling algorithm Variance Honey Bee Behavior with Multi-Objective Optimization (VHBBMO) has been proposed in [37] based on an ABC. Available jobs are assigned amongst the high- end servers. To reduce the makespan even more, the load between the high-end servers is also balanced. The scheduler should make job scheduling characteristics if a decision is made to balance the job. To balance the load, one job is eliminated from overloaded end servers and migrated to the appropriate underloaded end servers. The migration of a job depends on its priority level. There are three distinct groups for job-associated priorities, namely, medium, low, and high. Because of this, makespan and response time are better compared to ACO and particle swarm optimization. However, Power consumption factor and QoS factors are still disregarded for select proper underloaded end server (host or vim) to process the eliminated task from the overloaded end server.

In [12], Patel and Bhalodia (2019) integrated two algorithms to create a new algorithm for load balancing over cloud systems. This involves the identification of overloaded and underloaded VMs, followed by priority-based migration of tasks from overloaded to underloaded VMs. If priority is confirmed, the honeybee-inspired load balancing algorithm should be applied for task allocation, whereas if priority is not confirmed, the WRR algorithm should be adopted for task allocation. The modified honey bee algorithm outperforms the shortest job first and WRR algorithms in terms of CPU processing time and completion time. However, additional QoS factors (e.g., processing time, cost, power consumption) are not taken into account for select proper VM. Furthermore, the authors who proposed this method did not address determinants of performance (e.g., degree of imbalance, number of migrated tasks).

In [20] ,Kruekaew and Kimpan (2020) proposed a new method of heuristic task scheduling with ABC (HABC) by integrating the swarm intelligence algorithm with ABC and heuristic scheduling algorithms. The goal of this algorithm is to reduce makespan and achieve load balancing by providing better VM scheduling for cloud computing. Prior to applying the ABC algorithm, the First Come First Serve (FCFS), Smallest Job First, and Largest Job First heuristic algorithms were employed to structure the tasks into three distinct arrangements, with the arrangement using minimal computation time during task processing by the ABC algorithm considered the ideal. In terms of scheduling and load balancing, HABC with the largest job first heuristic algorithm (HABC-LJF) performed best. The key QoS factor identified was makespan, while other QoS factors and power consumption were not taken into account.

To determine the best way of allocating resources for each task in the dynamic cloud system so as to ensure that resources were neither overused nor underused, the Load balancing Extended Multi-objective Honeybee (LB-EMHB) algorithm was formulated by Alamelu et al. (2020) in [38]. The aim of the algorithm involves allocating an incoming task to a VM that satisfies the requirements and that performs fewer tasks than other VMs. Furthermore, none of the VMs can deviate from the average processing time by more than the pre-established threshold value. VM overloading occurs when that threshold value is exceeded. In this method, the tasks from overloaded VMs are eliminated and allocated to underloaded VMs that are most appropriate. The task in VMs' queues are prioritized as the task having minimal priority are chosen to be shifted to the underloaded VM. In comparison to ACO, the makespan has improved. LB-EMHB did not offer any novel addition as the primary aim and the underlying concept did not differ much from the one proposed by Hashem et al. (2017) in [22] . However, neither the degree of imbalance nor the number of tasks migrated was determined. Additional QoS factors and power consumption are not considered when choosing the best underloaded VM.

As noted by Joshi and Munisamy (2020) in [1], in the load balancing algorithm, the load is balanced through task scheduling and resource allocation. Task scheduling involves organizing the incoming requests (tasks) in a particular way to ensure appropriate utilization of the available resources [9]. In resource allo- cation, available resources are allocated to the required cloud applications online. Task scheduling and resource allocation are both non-polynomial hard optimization problems as there is a drastic difference in the number as well as length of tasks [1].

This makes it difficult to assess and determine optimal resource mapping. Joshi and Munisamy (2020) presented the Dynamic Degree Balanced with Membership value based (D2B-Membership) algorithm, which takes into consideration every host's membership value and the balance condition of VMs to modify the allocation policy. VMs are allocated to an appropriate host with more capacity than the VM's needs. Workload in task allocation is dynamically dispersed by assessing the load on a VM so that the system's performance can be improved. The results indicated that the algorithm improved upon RR and FCFS in terms of execution time as well as makespan. However, the algorithm did not observe the number of tasks migrated. The selection of the host and VM based on load conditions, without consideration for QoS factors and power consumption, is a drawback of the D2B-Membership algorithm.

4.2 Honeybee Behavior Algorithms Based on Power Consumption

Recently, the IT industry has significantly increased its energy use. As part of the endeavor to provide support to CC and grid computing services, major IT developers (e.g., IBM, Microsoft, Google) have established a greater number of data centers, which are pivotal hubs of information and communication technology [39] . Due to the myriad servers and switches they contain, these data centers consume a massive amount of energy, which not only makes operations more expensive but is also environmentally detrimental because of carbon dioxide emissions [40]. Energy consumption is further increased by the cooling equipment necessary to manage the heat generated by the datacenters [40]. It is estimated that, in 2012, data centers consumed about 300–400 TeraWatt Hour (TWh) of electricity (approximately 2% of global consumption), which is expected to triple by 2020 [39]. There is evidence that the energy used by idle servers is up to 70% of the energy consumed during peak activity [41].

As such, allowing servers to run with a limited workload is not cost effective. In the context of CC, a major problem for service providers is the amount of power used by cloud data centers [9]. In this regard, the goal is to decrease power use and ensure compliance with environmental policies and standard inter-user contracts [39]. A number of studies have investigated the power consumption of CC, with estimation and improvement of the power use of separate VMs.

As noted by Ghafari et al. (2013) [14], it is possible to decrease the amount of power used by cloud computing infrastructures with the load balancing method known as the ABC algorithm–minimal migration time (Bee-MMT), which uses ABC to identify over-utilized hosts. It then uses the minimum migration time policy for VM selection. The VM selection policy chooses one or more VMs to migrate from the over-utilized hosts so that their use can be minimized. It can also identify underutilized hosts and transfer all VMs assigned to these hosts, if possible, after which they are switched to sleep mode. The power and resource utilization are all minimized, as is evident from the results. Despite this, complexity and QoS were not adequately considered. Bee-MMT uses VM migration to achieve load balancing. Multiple VMs may be created on a single physical machine using virtualization. The VMs created will be independent in nature and have differing configurations. In the case of a physical machine becoming overloaded, it is necessary for VMs to transfer to a different host using a VM migration load

balancing approach [21]. However, this method has some disadvantages, the most significant of which are as follows [28] :

- A greater amount of memory is consumed by the physical machine from which the VM is migrated to the physical machine when the migration happens.
- Some of the customer's current activities can be lost, which can result in extremely high expenses.
- There may be excessive VM downtime as a result of halting VM migration.

To decrease the power consumption, the Bee-MMT is used; therefore, it will have more SLA violations compared to other approaches. Minimizing the SLA violation has become the main objective of the authors [42]. This is done as an obligation to meet the high standard of the QoS for customers. Auto- mated monitoring and the analysis of SLA using the constraint satisfaction issue are achieved based on the SALMonADA model.

A different strategy for the management of the power consumption of CC data centers, namely, the power-aware load balancing method Imperialism Competitive Algorithm-Minimum Migration Time (ICA-MMT), has been put forth by [43]. This method is based on the imperialism competitive algorithm for the identification of excessively used hosts and reduction of the use of those hosts by moving one or more of their VMs to other hosts. Meanwhile, every VM of underused hosts is moved to other hosts so that the underused hosts can be placed into sleep mode. According to the findings obtained, this method reduces power use and prevents SLA breach more effectively than other methods based on resource distribution, including local regression-minimum migration time, median absolute deviation-minimum migration time , and Bee-MMT, as well as methods without power awareness.

One method that has recently attracted significant research attention is VM consolidation, which can achieve a substantial decrease in power consumption, given that the amount of power consumed by inactive or sleep-mode hosts is low [5]. The VM consolidation method successfully reduces power consumption through the prevention of idle power consumption by changing idle nodes to modes such as sleep or hibernation, which use little power [44]. Live migration is also made possible by virtualization and involves VM transfer among hosts, known as physical servers or nodes, with minimal downtime. The overall goal is to ensure that the fewest possible nodes are active at any given moment [44].

In the VM consolidation method, choosing the VM for migration is no easy task, which has prompted the proposal of a range of solutions. A number of criteria must be considered in relation to this choice, given the dynamic nature of computation needs in the real world [45]. In [45], Monil and Rahman (2016) suggested a fuzzy VM selection algorithm with migration control. This method is capable of smart decision-making regarding the choice of VM for migration between two hosts. An overload detection algorithm was subsequently applied to establish mean, median, and standard deviation. The suggested method was simulated and compared with other similar methods and demonstrated better performance not only in reduced power consumption but also in minimizing SLA violations.

VM consolidation was the basis of the approaches implemented by [44] and [46] . Although both groups of researchers proposed fuzzy VM selection as a method for choosing a VM for migration from a host with excessive load, the inputs employed in a fuzzy inference system to decrease power consumption were different. More specifically, Monil and Rahman (2017) in [44] used RAM, correlation, and standard deviation as the inputs, Rajagopal and Baskaran (2019) in [46] used CPU and disk storage as inputs .

The studies cited above sought to decrease the power consumption by data centers or hosts by applying algorithms to move VMs or put idle nodes into a mode with low power consumption. Nevertheless, the strategy of VM migration does not address the issue of significantly diminishing power consumption. Furthermore, the method is deemed to lack efficiency if the reduction in the power consumption of data centers causes SLA violation or disregards service quality factors. Therefore, further investigation is necessary to overcome this issue.

5. SUMMARY OF PREVIOUS STUDIES

Many studies have focused on improving only a couple of factors among a multitude of QoS factors, without considering power consumption ([1],[11],[22],[35],[33]). On the other hand, various studies have presented load balancing algorithms based on ABC to reduce power consumption without considering QoS factors ([14]; [42]; [43]; [44]). None of those algorithms took into account the integration of QoS factors and power consumption in determining the host and the appropriate VM to receive the incoming task. Further, most previous works have relied on task migration for load balancing in systems. As a result, the load balancing process in previous algorithms starts only when the whole system is unbalanced, and it can have an effect on the degree of imbalance.

Table 2: The primary disadvantages and advantages of ABC-based load balancing

Ref.	Approach	Advantage	Disadvantage	Environment	Decision parameter
[30]	ABC	-Enhanced scalability and throughput.	-High response time.	Repast.NET	Throughput
[31]	A mutation operator within ABC	-Improved execution time .	-Overlooked other QoS factors.	-	Execution time.
[14]	Bee-MMT	-Power consumption is considered	More SLA violations	CloudSim toolkit	Power consumption and SLA violation
[24]	HBB-LB	-Enhance the makespan and response time.	-Constant occurrence of low priority tasks. -The degree of imbalance is high.	CloudSim toolkit	Response time, makespan, and degree of imbalance
[34]	Preemptive task scheduling based on ABC	-Efficient usage of resources.	-The number of task migration and the degree of imbalance are not considered.	CloudSim toolkit	Execution time.
[35]	HBBLBP	- Improved both execution time and cost.	-Considers cost as the main QoS parameter. -The degree of imbalance is high.	CloudSim toolkit	Degree of imbalance, cost, and execution time.
[10]	Improved version of HBB-LB	Makespan is low for both dependent as well as independent tasks	-The selection of VM based on load conditions only.	CloudSim toolkit	Makespan
[47]	An enhanced version of HBBLB.	-Improving the response time and processing time.	-Power consumption and QoS are disregarded .	Cloud analyst toolkit	Response time and processing time.
[22]	LBA_HB algorithm	-Improved both execution time and average response time.	-Power consumption and QoS are disregarded	CloudSim toolkit	Response time, degree of imbalance, and execution time .
[37]	VHBBMO	-Makespan and response time are enhanced	--Power consumption and QoS are disregarded	MATLAB	Makespan and response time.
[11]	IHBB-LB approach	- The number of task migration is reduced. Increased number of QoS parameters	-power consumption is not considered. -Degree of imbalance is high.	CloudSim toolkit	Makespan, degree of imbalance, and number of tasks migrated.
[34]	IE-ABC	-It helps in reducing the execution time	-power consumption is not considered.	CloudSim toolkit	Execution time and number of task migration
[12]	Modified honey bee behavior	-Improved processing time and completion time.	-Power consumption and QoS are disregarded .	CloudSim toolkit	Processing time and completion time.
[20]	HABC	-Minimize makespan	-Considered the makespan to be a major factor in QoS factors and overlooked power	CloudSim toolkit	Standard deviation and makespan.
[38]	LB-EMHB	Minimize the makespan	Power consumption and QoS are disregarded.	CloudSim toolkit	Makespan
[1]	D2B-Membership	Improved execution time and makespan.	Selection of the host and VM based on load conditions.	CloudSim toolkit	Makespan, execution time, and Throughput

6. CONCLUSION

Cloud computing can be viewed as a innovative paradigm that gives clients a great many assets to help with the synchronous execution of tasks.On other hand, the numerous applications and the continually changing client requests make issues of load balancing, including underloading and over-loading for the VMs in a cloud server .This adversely affects the system's performance.

In this paper, we gathered that many papers about a honey bee algorithm which are utilized for development in load Balancing process for cloud computing. In the introduction section we discuss about cloud computing, type and main components of cloud computing. Then explored the concept of load balancing, including its main types, and objectives, discussing the idea of a load balancing model and the various load balancing QoS metrics.

Honey bee algorithm is utilized for enhancment in cloud computing for development in QOS factors, energy consumption, task scheduling and load balancing. After overview study about Honey bee algorithm using for improvement in load balancing technique .We summarize the following conclusions :

- All the previous research did not reach the optimal results and the field is still wide to perform more and more experiments in the future to enhance and improve the services of CC.
- Further, recent studies have focused on energy-aware scheduling because data centers are known to be energy-hungry and have become a significant source of CO2 emissions. Reducing the data centers' energy consumption while ensuring that performance is not affected, and SLA constraints are not violated remains a challenge.
- Most studies that have used fuzzy logic have been focused on resource optimization, scheduling, and service based on cloud computing ability. However, these studies have failed to consider security in cloud computing in terms of fuzzy logic. It is thus necessary for future studies to address the security problem.
- Most of the previous research tested their results in a simulated environment such as a CloudSim simulator for this it is possible to apply the algorithms in a real-time environment to compare results.
- To effectively balance the load, more machine learning or deep learning-based approaches could be created.

REFERENCES

1. S. Chen, B. Mulgrew, and P. M. Grant. **A clustering technique for digital communications channel equalization using radial basis function network**s, *IEEE Trans. on Neural Networks*, Vol. 4, pp. 570-578, July 1993.
2. A. Joshi and S. D. Munisamy, **"Enhancement of Performance Parameter of Cloud Using Dynamic Degree Balanced with Membership Value Algorithm,"** Int. J. Adv. Res. Eng. Technol., vol. 11, no. 8, pp. 664–676, 2020, doi: 10.34218/IJARET.11.8.2020.065.
3. H. S. Alatawi and S. A. Sharaf, **"Hybrid Load Balancing Approach based on the Integration of QoS and Power Consumption in Cloud Computing,"** Int. J. Adv. Trends Comput. Sci. Eng., vol. 10, pp. 2278–3091, 2021.
4. G. Muthsamy and S. Ravi 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, 2020, doi: 10.1002/cae.22236.
5. P. Kumar and R. Kumar, "**Issues and challenges of load balancing techniques in cloud computing: A survey,"** ACM Comput. Surv., vol. 51, no. 6, 2019, doi: 10.1145/3281010.
6. E. H. Houssein, A. G. Gad, Y. M. Wazery, and P. N. Suganthan, **"Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends,"** Swarm Evol. Comput., vol. 62, no. May 2020, p. 100841, 2021, doi: 10.1016/j.swevo.2021.100841.
7. A. Ullah, N. M. Nawi, and M. H. Khan, **"BAT algorithm used for load balancing purpose in cloud computing: an overview,"** Int. J. High Perform. Comput. Netw., vol. 16, no. 1, p. 43, 2020, doi: 10.1504/ijhpcn.2020.110258.
8. S. T. Milan, L. Rajabion, H. Ranjbar, and N. J. Navimipour, **"Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments,"** Comput. Oper. Res., vol. 110, pp. 159–187, 2019, doi: 10.1016/j.cor.2019.05.022.
9. M. A. Elmagzoub, D. Syed, A. Shaikh, N. Islam, A. Alghamdi, and S. Rizwan, **"A survey of swarm intelligence based load balancing techniques in cloud computing environment,"** Electron., vol. 10, no. 21, 2021, doi: 10.3390/electronics10212718.
10. H. J. Shilpa Mehta, **"Load Balancing in Cloud Computing: A Comprehensive Survey on Recent Techniques,"** Int. J. Adv. Trends Comput. Sci. Eng., vol. 10, pp. 831 – 844, 2021, doi: 10.4018/978-1-7998-1021-6.ch016.
11. S. K. V. Geethu Gopinath, **"A Novel Improved Honey Bee Based Load Balancing Technique in Cloud Computing Environment,"** Asian Tournal Inf. Techonology, pp. 1425–1430, 2016.
12. M. A. S. Mosleh and G. Radhamani, **"A novel fuzzy QoS based Improved Honey Bee Behavior algorithm for efficient load balancing in cloud,"** SPIIRAS Proc., vol. 2, no. 57, pp. 26–44, 2018, doi: 10.15622/sp.57.2.
13. K. D. Patel and T. M. Bhalodia, **"An efficient dynamic load balancing algorithm for virtual machine in cloud computing,"** 2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019, no. Iciccs, pp. 145–150, 2019, doi: 10.1109/ICCS45141.2019.9065292.

14. A. Ullah, N. M. Nawi, J. Uddin, S. Baseer, and A. H. Rashed, **"Artificial bee colony algorithm used for load balancing in cloud computing: Review,"** IAES Int. J. Artif. Intell., vol. 8, no. 2, pp. 156–167, 2019, doi: 10.11591/ijai.v8.i2.pp156-167.
15. S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, **"Bee-MMT: A load balancing method for power consumption management in cloud computing,"** 2013 6th Int. Conf. Contemp. Comput. IC3 2013, no. April 2016, pp. 76–80, 2013, doi: 10.1109/IC3.2013.6612165.
16. A. Soni, G. Vishwakarma, and Y. Kumar Jain, **"A Bee Colony based Multi-Objective Load Balancing Technique for Cloud Computing Environment,"** Int. J. Comput. Appl., vol. 114, no. 4, pp. 19–25, 2015, doi: 10.5120/19967-1825.
17. M. Gamal, R. Rizk, H. Mahdi, and B. Elhady, **"Bio-inspired load balancing algorithm in cloud computing,"** Adv. Intell. Syst. Comput., vol. 639, pp. 579–589, 2018, doi: 10.1007/978-3-319-64861-3_54.
18. O. F. Aloufi, K. Djemame, F. Saeed, and F. Ghaban, **"A survey on predicting workloads and optimizing QoS in the cloud computing,"** 2021 Int. Congr. Adv. Technol. Eng. ICOTEN 2021, 2021, doi: 10.1109/ICOTEN52080.2021.9493436.
19. Et. al., Ibrahim Mahmood Ibrahim, **"Task Scheduling Algorithms in Cloud Computing: A Review,"** Turkish J. Comput. Math. Educ., vol. 12, no. 4, pp. 1041–1053, 2021.
20. K. Balaji, P. Sai Kirani, and M. Sunil Kumar, **"Load balancing in Cloud Computing: Issues and Challenges,"** Turkish J. Comput. Math. Educ., vol. 12, no. 2, pp. 3077–3084, 2021, doi: 10.14419/ijet.v7i1.1.9709.
21. B. Kruekaew and W. Kimpan, **"Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing,"** Int. J. Comput. Intell. Syst., vol. 13, no. 1, pp. 496–510, 2020, doi: 10.2991/ijcis.d.200410.002.
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. 654–661, 2019, doi: 10.1145/3281010.
23. W. Hashem, H. Nashaat, and R. Rizk, **"Honey bee based load balancing in cloud computing,"** KSII Trans. Internet Inf. Syst., vol. 11, no. 12, pp. 5694–5711, 2017, doi: 10.3837/tiis.2017.12.001.
24. 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., 2021, doi: 10.1016/j.jksuci.2021.02.007.
25. L. D. Dhinesh Babu and P. Venkata Krishna, **"Honey bee behavior inspired load balancing of tasks in cloud computing environments,"** Appl. Soft Comput. J., vol. 13, no. 5, pp. 2292–2303, 2013, doi: 10.1016/j.asoc.2013.01.025.
26. Z. Zhou, H. Wang, H. Shao, L. Dong, and J. Yu, **"A high-performance scheduling algorithm using greedy strategy toward quality of service in the cloud environments,"** Peer-to-Peer Netw. Appl., vol. 13, no. 6, pp. 2214–2223, 2020, doi: 10.1007/s12083-020-00888-4.
27. M. H. Ghahramani, M. Zhou, and C. T. Hon, **"Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services,"** IEEE/CAA J. Autom. Sin., vol. 4, no. 1, pp. 6–18, 2017, doi: 10.1109/JAS.2017.7510313.
28. H. Talebian et al., **Optimizing virtual machine placement in IaaS data centers: taxonomy, review and open issues,** vol. 23, no. 2. Springer US, 2020.
29. G. Megharaj and G. M. Kabadi, **Metaheuristic-Based Virtual Machine Task Migration Technique for Load Balancing in the Cloud**, vol. 771. Springer Singapore, 2019.
30. M. R. Thanka, P. Uma Maheswari, and E. B. Edwin, **"An improved efficient: Artificial Bee Colony algorithm for security and QoS aware scheduling in cloud computing environment,"** Cluster Comput., vol. 22, pp. 10905–10913, 2019, doi: 10.1007/s10586-017-1223-7.
31. J. Yao and J. H. He, **"Load balancing strategy of cloud computing based on artificial bee algorithm,"** Proc. - 2012 8th Int. Conf. Comput. Technol. Inf. Manag. ICCM 2012, vol. 1, pp. 185–189, 2012.
32. M. Gupta and G. sharma, **"An Efficient Modified Artificial Bee Colony Algorithm for Job Scheduling Problem,"** Int. J. Soft Comput. Eng., no. 1, pp. 2231–2307, 2012.
33. K. R. Remesh Babu and P. Samuel, **"Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud,"** Adv. Intell. Syst. Comput., vol. 424, pp. 67–78, 2016, doi: 10.1007/978-3-319-28031-8_6.
34. S. M. Mousavi and F. Gábor, **"A novel algorithm for Load Balancing using HBA and ACO in Cloud Computing environment,"** Int. J. Comput. Sci. …, 2016, [Online]. Available: https://www.academia.edu/download/47321119/07_Paper_31051620_IJCSIS_Camera_Ready_A_48-52.pdf.
35. T. Nadu, **"Nature Inspired Preemptive Task Scheduling for,"** no. 978, pp. 0–5, 2014.
36. J. S. Sheeja Y S, **"Cost Effective Load Balancing Based on honey bee Behaviour in Cloud Environment,"** First Int. Conf. Comput. Syst. Commun., vol. 13, no. December, pp. 214–219, 2014.
37. M. S. George, K. C. Nithin Das, and B. R. Pushpa, **"Enhanced honeybee inspired load balancing algorithm for cloud environment,"** Proc. 2017 IEEE Int. Conf. Commun. Signal Process. ICCSP 2017, vol. 2018-Janua, pp. 1649–1653, 2018, doi: 10.1109/ICCSP.2017.8286670.
38. N. Sethi, S. Singh, and G. Singh, **"Multiobjective Artificial Bee Colony based Job Scheduling for Cloud Computing Environment,"** Int. J. Math. Sci. Comput., vol. 4, no. 1, pp. 41–55, 2018, doi: 10.5815/ijmsc.2018.01.03.
39. M. Alamelu, **"Enhanced Multi-Objective based Resource Allocation using Framework Creation in**

Cloud Computing," Int. J. Res. Appl. Sci. Eng. Technol., vol. 8, no. 6, pp. 1303–1309, 2020, doi: 10.22214/ijraset.2020.6210.

40. M. Soltanshahi, R. Asemi, and N. Shafiei, **"Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers,"** Heliyon, vol. 5, no. 7, pp. 3–8, 2019, doi: 10.1016/j.heliyon.2019.e02066.
41. N. Chaurasia, M. Kumar, R. Chaudhry, and O. P. Verma, **Comprehensive survey on energy-aware server consolidation techniques in cloud computing,** vol. 77, no. 10. Springer US, 2021.
42. S. Ismaeel, R. Karim, and A. Miri, "**Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres,"** J. Cloud Comput., vol. 7, no. 1, 2018, doi: 10.1186/s13677-018-0111-x.
43. S. Saravanan, V. Venkatachalam, and S. T. Malligai, **"Optimization of SLA Violation In Cloud Computing Using Artificial,"** vol. 1, no. 3, pp. 410–414, 2015.
44. S. Yakhchi, S. . Ghafari, M. Yakhchi, M. Fazeli, and A. Patooghy, **"ICA-MMT: A load balancing method in cloud computing environment.,"** 2015, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7210303/.
45. M. A. H. Monil and R. M. Rahman, **"Fuzzy logic-based VM selection strategy for cloud environment,"** Int. J. Cloud Comput., vol. 6, no. 2, pp. 163–186, 2017, doi: 10.1504/IJCC.2017.086019.
46. M. A. H. Monil and R. M. Rahman, **"VM consolidation approach based on heuristics fuzzy logic, and migration control,"** J. Cloud Comput., vol. 5, no. 1, 2016, doi: 10.1186/s13677-016-0059-7.
47. E. Rajagopal and N. Baskaran, **"Fuzzy softset based VM selection in cloud datacenter,"** 2019 Int. Conf. Intell. Comput. Control Syst. ICCS 2019, no. Iciccs, pp. 462–467, 2019, doi: 10.1109/ICCS45141.2019.9065678.