ISSN 2278-3091
Volume 10, No.2, March - April 2021
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse841022021.pdf
https://doi.org/10.30534/ijatcse/2021/841022021

# Hybrid Load Balancing Approach based on the Integration of QoS and Power Consumption in Cloud Computing

**Hind Salem Alatawi[1],Sanaa Abdullah Sharaf[2]**
[1]Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia,halatawi006@stu.kau.edu.sa
[2]Computer Science Department, King Abdulaziz University, Jeddah, Saudi Arabia, sshara@kau.edu.sa

## ABSTRACT

Over the past few years, there has been keen research interest in load balancing and task scheduling in the cloud as the extensive amount of data that is stored in the server leads to significantly increased load. This can be resolved by using a hybrid algorithm in which the honeybee behavior algorithm's advantages are integrated with fuzzy logic to conduct task scheduling and as well as balancing in the cloud. The design of this hybrid algorithm aims to enhance prior approaches. It is developed as per ABC and merges the important QoS factors along with power consumption so that the power that virtual machines (VMs) consume on the host can be precisely assessed, thereby ensuring efficient load balancing algorithm. The present study aims to evaluate the VMs' power consumption by taking into account crucial QoS factors for selecting which host and virtual machine will be best suited for receiving the task. CloudSim was used to simulate the ILBA_HB algorithm. In terms of makespan, average response time, and degree of imbalance, the performance of the ILBA HB algorithm is compared to that of the LBA HB and HBB-LB algorithms. According to the results, the proposed algorithm outperformed LBA_HB and HBB-LB.

**Key words :**Cloud Computing, Load Balancing, Honeybee approach, Fuzzy logic.

## 1. INTRODUCTION

Cloud Computing (CC) is an emerging computing model encompassing multiple technologies such as parallel computing, utility computing, and distributed computing [1]. The CC concept encourages users to share computing resources and data through a host application service provider, eliminating the need for users to pay for energy or purchase a server [2]. However, as the number of users who want to participate in CC grows, so does the demand for shared resources, making it more difficult for hosts to load balance between different resources and schedule tasks appropriately [3].

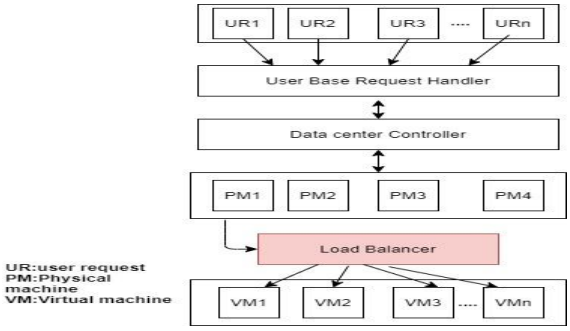


**Figure 1:**Model for load balancing [4]

Load balancing is required in CC for virtual machines (VM) to handle both overload and underload conditions [5]. Figure 1 presents a model and workflow of such a load balancer [4]. The dynamic nature of CC thus requires the application of dynamic algorithms to allow hosts to load balance effectively [6]. Due to the limitations on conventional load balancing algorithms in light of changing workload dynamics [3]. Swarm Intelligence algorithms (SI), such as the artificial bee colony (ABC) and ant colony optimization (ACO), have been developed in recent years to help solve such challenges [7].

Several researchers have proposed SI-based algorithms for balancing loads in the cloud, including the ABC algorithm mentioned above [8]. The Load Balancing Algorithm based on Honeybee behavior (LBA_HB) is an example of an ABC algorithm in which load balancing is based on just one factor from among the relevant quality of service factors (QoS); other important factors are thus seemingly ignored during load balancing in this case. The first objective is to select a suitable host with minimal processing time and the second objective selects a suitable VM to process the incoming task using a minimum request count. This avoids the challenges with other current ABC algorithms in that they disregard the power consumption factor of the VM, tend not to reduce to degree of imbalance in the cloud, and incur a large number of task migrations.

This study proposes an improved load-balancing algorithm (ILBA HB) based on honeybee behavior that considers QoS parameters such as processing time, cost, and power consumption. ILBA_HB designs two fuzzy inference systems that allow for selection of the proper host followed by

selection of the appropriate VM based on power consumption and QoS factors. The performance of the proposed ILBA HB algorithm is compared to that of the basic LBA HB algorithm in terms of average response time, makespan, and degree of imbalance using CloudSim simulator.

This paper is organized as follows. We present related work in the following section. Section 3 introduces the proposed load-balancing algorithm. The experimental results and evaluation of the proposed algorithm are presented in Section 4. Section 5 discusses the findings and future work.

## 2. RELATED WORK

Countless clients share cloud services by sending their processing tasks to cloud environments. One challenge that this presents to a cloud server is the scheduling of millions of tasks. Studies in the SI field have shown that complicated issues could be resolved if similar agents worked together in clusters within a cloud server [3]. Effective scheduling and load balancing can help ensure that customers receive enhanced QoS [9]. This section reviews load-balancing algorithms inspired by artificial bee colony .

In [10], authors developed the honeybee-behavior–based load-balancing (HBB-LB) algorithm, which enables load balancing and considers the priorities of tasks taken away from VMs with heavy loads. 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 Weighted Round Robin (WRR), First In First Out (FIFO), 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 by this algorithm.

When selecting a VM, the HBB-LB algorithm considers only the load conditions and ignores other critical QoS factors required to improve cloud computing performance.As a result, as shown in[8]an improved honey bee behavior based load balancing (IHBB-LB) algorithm was developed. More QoS parameters of the VM, such as service response time and cost, were added to improve load balancing. Results illustrate that the IHBB-LB algorithm outperformed the HBB-LB algorithm in terms of makespan, response time, and the number of migrated tasks. The degree imbalance for HBB-LB and IHBB-LB, according to the results, was 1.45 and 1.43, respectively, indicating a slight increase in the DI associated with IHBB-LB.

In [11], 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. However, additional QoS factors (e.g., response time, cost, power consumption) are not taken into account. Furthermore, the authors who proposed this method did not address determinants of performance (e.g., degree of imbalance, number of migrated tasks).

In [12], Kruekaew and Kimpan (2020) proposed a new method of heuristic task scheduling with ABC (HABC) by integrating the swarm intelligence algorithm with ABCand heuristic scheduling algorithms. The goal of this algorithm is to reduce makespan and achieve load balancing by providing better VM scheduling for cloud computing in both uniform and non-uniform media. Prior to applying the ABC algorithm, the First Come First Serve (FCFS), Smallest Job First (SJF), and Largest Job First (LJF) 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 use were not taken into account.

In [13], 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 Round Robin (RR) and FCFS in terms of execution time as well as makespan. However, the algorithm did not compute the degree of system imbalance or the number of tasks migrated. The algorithm was concerned with the two factors of makespan and execution time, regarding them as the major QoS factors, and ignored power consumption by either the VM or host.

Recently, the IT industry has significantly increased its energy use. As part of the endeavor to provide support to cloud computing 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 [14]. 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. Energy consumption is further increased by the cooling equipment necessary to manage the heat generated by the datacenters

[15]. It is estimated that, in 2012, data centers consumed about 300–400 TWh of electricity (approximately 2% of global consumption), which is expected to triple by 2020 [16]. There is evidence that the energy used by idle servers is up to 70% of the energy consumed during peak activity [14], [17].

As such, allowing servers to run with a limited workload is not cost effective. In the context of cloud computing, a major problem for service providers is the amount of power used by cloud data centers [14], [18]. In this regard, the goal is to decrease power use and ensure compliance with environmental policies and standard inter-user contracts [14]. A number of studies have investigated the power use of cloud computing, with estimation and improvement of the power use of separate VMs.

As noted in [19] by Ghafari, Fazila, Patooghy, and Rikhtechi (2013), 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 (MMT) 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. As can be seen from the performance, power and resource consumption are both reduced.Regardless, QoS factors were not properly considered.To accomplish load balancing, Bee-MMT employs VM migration. 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 [4]. However, this method has some disadvantages, the most significant of which are as follows [20]:

- When the VM is migrated from one physical machine to another, the physical machine that the VM is migrated to consumes more memory.
- 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.

One method that has recently attracted significant research attention is VM consolidation, which can achieve a substantial decrease in power use, given that the amount of power consumed by inactive or sleep-mode hosts is low [21]. The VM consolidation method successfully reduces power use through the prevention of idle power use by changing idle nodes to modes such as sleep or hibernation, which use little power [22]. 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 [21].

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. In [21], 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 use but also in minimizing Service-Level Agreement (SLA) violations.

VM consolidation was the basis of the approaches implemented in [22] and [23]. 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 use were different. More specifically, Monil and Rahman (2017) used RAM, correlation, and standard deviation as inputs in [22], while Rajagopal and Baskaran (2019) used CPU and disk storage as inputs in [23] used CPU and disk storage as inputs.

The studies cited above sought to decrease power use 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 use. Furthermore, the method is deemed to lack efficiency if the reduction in the power use of data centers causes SLA violations or disregards QoS factors. Therefore, further investigation is necessary to overcome this issue.

Many studies have concentrated on improving just a few factors among a plethora of QoS factors[6], [8],[10],[13], [24], [25],[26]. Without considering power usage, such research has centered on improving QoS factors. On the other hand, numerous studies[19], [22], [23], [26], [27]have proposed load-balancing algorithms based on ABC to minimize power consumption with ignoring QoS variables.

To evaluate the host and the required VM to receive the incoming task, all ABC-based load-balancing algorithms have ignored the integration of QoS factors with power consumption. Previous algorithms only began the load-balancing process if the whole system was unbalanced enough to influence the degree of imbalance. As a result, this research proposes an improved LBA HB algorithm for estimating the VM's power consumption and minimizing the system's imbalance degree, average response time, and makespan.

## 3. PROPOSED MODEL

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 [28]. ABC refers to a subsection of the swarm-intelligence based algorithms that address different optimization problems by imitating the honeybee swarms' collective intelligence [28]. 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 [10]. Sharing information helps with making decisions and examining the search space [8].

Employed bees, scouts, and onlookers are the three types of bees in an ABC system; each performs a different role[3] :

1. Employed bees: The employed bees are also called leader bees. 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 [25].

2. Scout bees: These bees search for new food sources at random near the hive, and when they come across an existing source, they begin a new search for a new source in that area[9].

3. Onlooker bees: Onlooker bees gather information from the employed bees in the hive and select food sources based on these dances [8]. Typically, a honeybee swarm is comprised of, on average, 50% employed bees and 50% unemployed bees, with 5 to 10% acting as scout bees. Both scouts and onlookers are also known as unemployed bees [25].

In this study, the methodology is based on the honeybees' foraging activity carried out to determine the most suitable available resources that can ensure that the cloud data center has load balancing. Similar to the activity of honeybees when searching for food sources wherein they select the best food source through foraging, the proposed algorithm approach determines the best suited VM in the appropriate host to allocate the tasks through the fuzzy inference system. Table 1 presents a mapping between the honeybees' foraging behavior and the proposed system's load balancing scenario obtained by conducting an in-depth analysis of the honeybees' foraging behavior.

ILBA_HB is an extension of LBA_HB that enhances the performance of LBA_HB with fuzzy logic. The ILBA_HB algorithm balances the load using a two-level approach. In the first level, the suitable host is selected based on capacity, processing time, and load. In the second level, the appropriate VM is selected based on processing time, and cost. The VM's power consumption on the host is also estimated. A fuzzy system is used to select a proper host and VM to process tasks. Fuzzy systems are suitable for environments wherein multiple factors are undetermined or cannot be predicted. One of the primary benefits of the fuzzy logic controller compared with other conventional control strategies is that its controlled development can be sustained without complex mathematical modeling. Due to the benefits of fuzzy logic when working in dynamic, uncertain environments, this paper proposes the fuzzy approach.

Previous algorithms have used two methods to balance the system: VM migration and task migration. The migration of VMs has several limitations, as previously mentioned. In contrast, task migration undertakes the transfer of novel or excess tasks from a VM with more load to a VM with less load. The completion time of migration depends upon the number of tasks migrated from a VM in the load-balancing process. The migration of several tasks from one virtual machine to another causes unfavorable load conditions in the VM from which the tasks are migrated. The occurrence of unfavorable load conditions must be minimized to efficiently balance the load. For this reason, the proposed algorithm balances the system load as soon as a user request arrives and assigns that request to the appropriate host and then the appropriate VM. That means the load-balancing process starts with the arrival of the system's first task, not after the whole system becomes unbalanced. Therefore, the imbalance degree is decreased in ILBA_HB. If all VMs on a host become overloaded, then that host becomes overloaded. ILBA_HB restricts the allocation of requests to overloaded hosts by removing the task from the waiting queue, where it must wait until it can be allocated to an available host. Therefore, ILBA_HB avoids migration time cost, which is caused by the migration of a number of tasks.

### 3.1 Description

Fuzzy VM controller and fuzzy host controller were planned as part of this research. However, since the situation is uncertain and computing requirements are highly complex in practice, fuzzy logic can be implemented at various levels of input to achieve the best combination of energy consumption and QoS factors[29].

**Table 1:**Mapping between the ILBA_HB and honeybees' behavior

| Honeybee Hive | Cloud Environment |
|---|---|
| Food Source | VMs on proper host, VM or Host, |
| Honeybee | Task (Cloudlet) |
| Bee selects best food source (onlooker bee) | Task allocated to proper VM by Honeybee broker |
| Scout bees | Tasks in queue/new incoming task |
| Employed bees | Tasks running in each VM |
| Bee finds depleted food source | VM is overloaded |
| Information shared through waggle dance | Load, capacity in host and processing time on each host and VM |

#### 3.1.1 Fuzzy Host Selection Method

For the fuzzy host controller system, three inputs have been specified: load, processing time, and capacity. The host's main characteristics are capacity and load. The host fuzzy controller produces fitness values for all hosts and then chooses the host with the highest fitness value. The output of the host fuzzy controller is the host with the highest fitness value. Figures 2, 3, 4, and 5 illustrate the membership functions for three inputs and one output. To provide a fuzzy host controller,a fuzzy inference rule (FIS) is developed using the three metrics as input. The FIS requires linguistic variables to be used in the generation of fuzzy inference rules[21]. Table 2 shows the six possible logical product and output response conclusions in the proposed method.

A CC system is comprised of a series of data centers. Each data center encompasses a series of h hosts, each of which in turn encompasses a series of m VMs. A set of VMs is created as$V = \{V_1, V_2, \dots, V_m\}$, where $m$ is the number of all VMs. A set of tasks is created as $T = \{T_1, T_2, \dots, T_n\}$, where $n$ is the number of all tasks. The completion time for a task$T_i$ on a $\mathrm{VM}_j$ is indicated as $CT_{ij}, i = 1, 2, \dots n$ , $j = 1, 2, \dots m$ .The

following equations are used to calculate these inputs:

- The total possible resources of a VM that can be allocated for task execution are determined by capacity. If $N_p$ denotes the number of processors in a virtual machine, $P_{mips}$ denotes the million instructions per second (MIPS) for all processors, and $B$ denotes the bandwidth available for communication, the capacity of a single virtual machine (Ci) can be determined as follows[12]:

$$Ci = N_p + P_{mips} + B \quad (1)$$

A host's capacity is the summation the capacity of all VMs, given as follows [8]:

$$C = \sum_{i=1}^{m} Ci \quad (2)$$

- If $T(t)$ represents the number of tasks on a VM's service queue at time t, , and S represents the execution time $S(T, t)$, then the load $L_{v,t}$ is computed as follows [13]:

$$L_{v,t} = \frac{T(t)}{S(T,t)} \quad (3)$$

The host's load can be obtained by summing the load of all VMs:

$$L = \sum_{j=1}^{m} L_{v,t} \quad (4)$$

- The processing time of a host is calculated as follows [3]:

$$PT = \sum_{j=1}^{m} PT_j \quad (5)$$

Similarly, where n denotes the number of hosts, the average processing time of all hosts can be determined as follows[3]:

$$PT_{AVG\_host} = \frac{1}{n}\sum_{i=1}^{n} PT \quad (6)$$

- Load standard deviation: The standard deviation (σ) in statistics and probability theory shows how much variance or dispersion there is in relation to the average. The formula is as follows[10]:

$$\sigma = \sqrt{\frac{1}{m}\sum_{j=0}^{m} (PT_j - PT_{AVG\_VM})^2} \quad (7)$$

- Standard normal deviate of VM(j): The calculated difference between an observed value and the mean is referred to as deviation in mathematics and statistics. The magnitude of the difference indicates the extent of the difference[3]. The normal deviate, also known as the standardized value, is the distance between the mean of a data point and the standard deviation of a distribution. The standard normal deviate is a unit deviation with a mean of zero. This metric is determined as follows to show deviation from the average mean or predicted value[3]:

$$SND_{VM}(j) = \frac{(PT_{VM}(j) - PT_{AVG\_VM})}{\sigma} \quad (8)$$

### 3.1.2 Fuzzy VM Selection Method

For the development of the fuzzy VM controller system, three inputs have been specified: processing time, cost, and power consumption.The VM fuzzy controller produces fitness values for all VMs on the proper host and then chooses the VM with the highest fitness value to process the incoming task. The output of the VM fuzzy controller is the VM with the highest fitness value. Figures 6, 7, 8, and 9 illustrate the membership functions for three inputs and one output. To provide a fuzzy VM controller, a FIS is developed using the three metrics as input. The FIS requires linguistic variables to be used in the generation of fuzzy inference rules [21]. Table 3 shows the seven possible logical product and output response conclusions in the proposed method.The following equations are used to calculate these inputs:

- The processing time of VMj is calculated as follows [8]:

$$PT_j = \frac{L_{v,t}}{Ci} \quad (9)$$

The following formula is used to measure the average processing time of all virtual machines:

$$PT_{AVG\_VM} = \frac{1}{m}\sum_{j=1}^{m} PT_{VM}(j) \quad (10)$$

- The cost of a virtual machine is calculated by the cost of a single CPU, network, RAM, and bandwidthunit. If a VM is priced such that p represents unit price where $p = 1$, $data$ represents bandwidth, net represents network units, RAM represents memory units, and $a, b, c, d$ represents weights for each resource attribute where $a + b + c + d = 1$ ,The VM's cost is then calculated as follows[8]:

$$Cost = \frac{p}{CPU^a * net^b * data^c * RAM^d} \quad (11)$$

- On a given host, a VM's power consumption can be measured as follows. Consider the $y$th VM, , denoted by $VM_y$, where $c_y$ denotes the number of CPU nodes in $VM_y$ . The memory capacity of $VM_y$ is $m_y$, the task within $VM_y$ is $t_{iy}$, and the CPU utilization ratio of $t_{iy}$ is $c_{iy}$. If $t_{iy}$ is run on multiple CPUs, $c_{iy}$ becomes the sum of the Processor utilization ratios for each of the CPUs where $t_{iy}$ is run[20]. $m_{iy}$ is the memory utilization of $t_{iy}$, and $p_{iy}$ is the consumed power of $t_{iy}$ .The CPU and memory usage of a task are essential factors in VM power consumption, and when these factors have higher values, VM power consumption increases[20].As shown in Eq. 12, $p_{iy}$ is estimated as the product of $c_{iy}$ and $m_{iy}$.

$$p_{iy} = c_{iy} \times m_{iy} \quad (12)$$

$p_y$, as shown in Eq. 16, represents the total consumed power of each task in $VM_y$, while $n$ denotes the overall number of tasks running in $VM_y$[20]

$$p_y = \frac{\sum_{j=1}^{n} p_{jy}}{c_y} \quad (13)$$

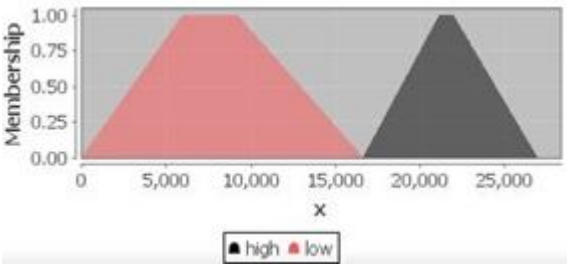


**Figure 2:**Capacity input membership function

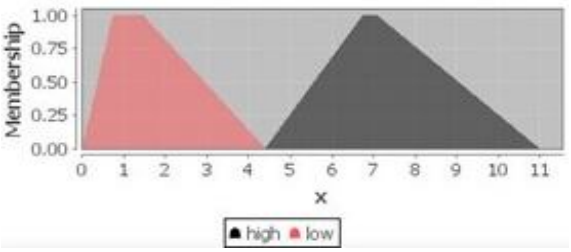


**Figure 3:**Processing timeinput membership function

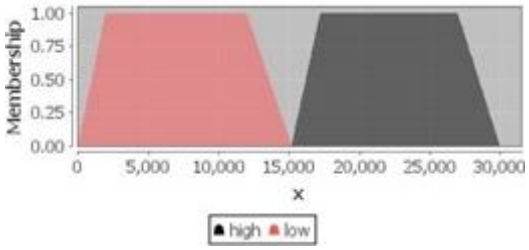


**Figure 4:**Loadinput membership function

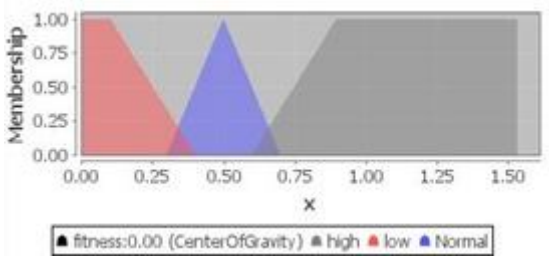


**Figure 5:**Membership output function of host-fitness

**Table 2:**The fuzzy rulers for hosts

| **The fuzzy rulers to host selector are:** |
|---|
| 1: IF capacity IS high AND load IS low AND processing_time IS low THEN fitness IS high. |
| 2: IF capacity IS low AND load IS high AND processing_time IS high THEN fitness IS low. |
| 3: IF capacity IS low AND load IS low AND processing_time IS low THEN fitness IS Normal. |
| 4: IF capacity IS high AND load IS low AND processing_time IS high THEN fitness IS Normal; |
| 5: IF capacity IS high AND load IS high AND processing_time IS low THEN fitness IS high; |
| 6: IF capacity IS high AND load IS high AND processing_time IS high THEN fitness IS low; |

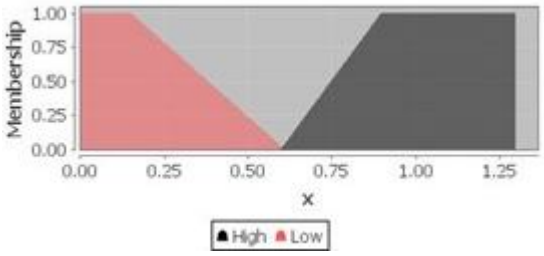


**Figure 6:**Cost input membership function

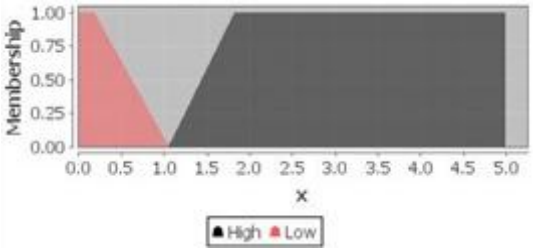


**Figure 7:** VM processing_time input membership function

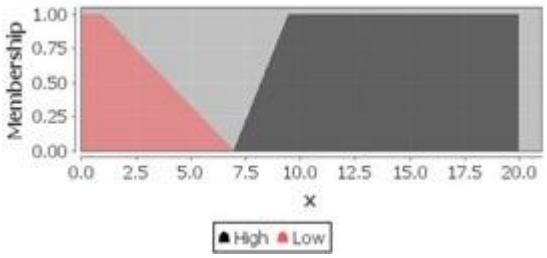


**Figure 8:**Power consumptioninput membership function

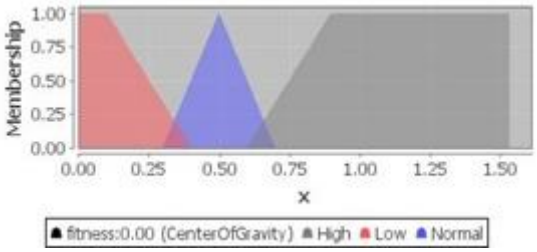


**Figure9:**Membership output function of VM-fitness

**Table 3:** The fuzzy rulers for VMs

| **The fuzzy rulers to VM selector are:** |
|---|
| 1: IF Power IS Low AND Cost IS Low AND VM_PTTime IS Low THEN fitness IS High; |
| 2: IF Power IS High AND Cost IS High AND VM_PTTime IS High THEN fitness IS Low; |
| 3: IF Power IS Low AND Cost IS High AND VM_PTTime IS Low THEN fitness IS High; |
| 4: IF Power IS High AND Cost IS Low AND VM_PTTime IS High THEN fitness IS Low; |
| 5: IF Power IS High AND Cost IS High AND VM_PTTime IS Low THEN fitness IS Normal; |
| 6: IF Power IS Low AND Cost IS Low AND VM_PTTime IS High THEN fitness IS Normal; |
| 7: IF Power IS Low AND Cost IS High AND VM_PTTime IS High THEN fitness IS Normal; |

### 3.2 Flowchart of Proposed Model

Whilst the four stages outlined below (A, B, C, & D) are comparable to the four stages described in LBA_HB (A, B, C, & D), there is one noticeable difference in that the suggested algorithm encompassed two further steps, to wit: B.8 and B.9. Figure 10 presents the framework of the ILBA_HB algorithm. The four phases in the algorithm are outlined below.

A. Workflow Submission (the input comprises a group of tasks)

1. The preprocessor receives new workflow.
2. The number of tasks and the length of each task's instructions are calculated in the workflow sent to the preprocessor.

B. Task In (input comprises an individual task)

1. Tasks are readied for implementation, and a request is dispatched to the VM load balancer.
2. Prepared tasks are queued by the VM load balancer.
3. Tasks are prioritized as per a FCFS approach and the queued first task is taken by the VM load balancer.
4. To determine which task is allocated to which VM, data is gathered between the last allocation and de-allocation, thereby imitating honeybee behavior since bees determine which plant to visit in accordance with nectar availability in specific flower patches. Threshold information gauges host availability in accordance with host variation. VMs perform this availability check.
5. Host levels are limited so request allocations to overloaded hosts are restricted. However, any task withdrawn from the queue must locate an appropriate host. The threshold data indicates two potential outcomes, namely: the task might locate a list of hosts using fuzzy host selection or the task might not find a host and be waited until one is obtained.
6. The first task is eliminated from the queue when it identifies a suitable host.
7. A control flag gathers tasks from the queue.
8. (Fuzzy host level): Three vital inputs are located in the fuzzy host selection, whereby the host is selected in accordance with fuzzy host selection output.
9. (Fuzzy VM level):It's possible that a task may be accepted by more than one VM on a given host. The most appropriate VM is chosen using a fuzzy VM selection method. The most appropriate VM is chosen based on three key inputs. A proper VM id is the output of fuzzy VM selection.

C. Task out (an appropriate VM has been selected):

1. The host and the VM update any allocated data. Regular checks designed to whether the host or VM is overloaded can produce data, including the host's and the VM's current processing time. Whether the VM(j) becomes overloaded is determined by variations in its processing time. The variation value indicates the VM(j) load balance in comparison to those of other VMs. Variations in VM processing time, when compared to average VM processing times indicate whether it exceeds or is equal to a threshold value α, as per the following:

$$SND_{VM}(j) \geq \alpha \qquad (14)$$

When all VMs in a specified host become overloaded, that host is considered overloaded.
2. The task is allocated to a suitable VM.
3. The task upgrades any de-allocated data, including current host and VM processing times and host and VM availability. VMs are deemed ready for use when specific VM processing times vary from the average processing time of all VMs at a rate lower than a threshold value α, as indicated in the following:

$$SND_{VM}(j) < \alpha \qquad (15)$$

A host which has at least one available VM is considered available.
4. Remaining tasks regarded both host and VM are conducted in a fashion akin to the waggle dance of honeybees, wherein information is passed to other bees in the hive. In the current case, information updates pertain to availability and VM and host loads, thereby facilitating the suitable assignment of task to VMs.
5. To avoid delays once the allocated task has been executed, a control lag is dispatched to the tasks.
6. The execution is completed by this task.

D. Delay (a task is unable to locate an available host)

1. The process is delayed unit the control flag has been dispatched and received, after which the execution can be concluded. Subsequently, the task is restarted from an earlier stage because the perceived data must be re-evaluated. Delayed tasks are prioritized ahead of other queuing tasks. Once the first round is completed, each task returns to phase B.4, after which is concludes the other unfinished steps.

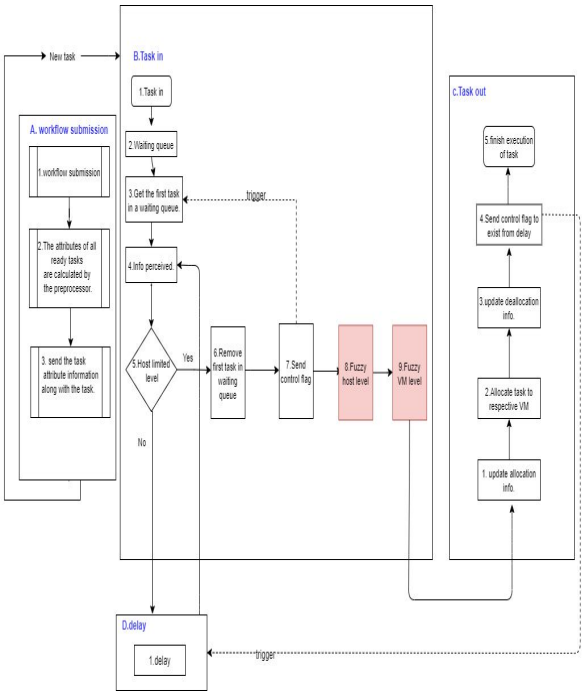


**Figure 10:**Flow chart of structure of ILBA_HB model

## 4. RESULTS AND DISCUSSION

We present the results of the experiments conducted in this study in this section. We compared the proposed algorithm with the HBB-LB and LBA-HB algorithms. We used the CloudSim simulator for execution. To mimic our algorithm, we expanded the classes of the CloudSim simulator. The principle execution is comprised of a subclass of the DatacenterBroker class in the org.cloudbus.cloudsim package, which we called HoneyBeeLoadbalancer. It is imperative to note that the DatacenterBroker class has an inheritance relationship with the SimEntity class within the org.cloudbus.cloudsim.core package.

To assess the performance in the cloud, results were simulated in a Windows 10 environment (64-bit) with an i5 processor and 2.3 GHz speed with 16 GB memory. The language used was Java (Eclipse).

In this study, we used the jFuzzyLogic library to combine CloudSim and fuzzy logic. This library includes a programming interface as well as a completely functioning and complete implementation of a FIS [22].

### 4.1Simulation Parameters

The proposed method's output is evaluated in a heterogeneous setting using VMs with varying characteristics. This cloud environment accepts cloudlets of various specifications. Table 4 summarizes the characteristics of a datacenter, VMs, hosts, as well as cloudlets. The simulation with the CloudSim framework follows the next steps:

1. Set up the CloudSim package
2. Create Datacenters
3. Create Broker
4. Create VMs and Cloudlets and send them to broker
5. Starts the simulation
6. Save results when simulation is over

**Table 4:** Simulation Parameters

| Type | Parameters | Values |
|---|---|---|
| DC | Number of Data Centers | 10 |
| | Number of Hosts | 2-6 |
| | VmScheduler | Time shared |
| | Host Processing speed (MIPS) | 10000 |
| | Host storage (MB) | 100000 |
| | Host bandwidth (Mbps) | 80000 |
| | Host RAM (MB) | 4096Mb |
| VM | Number of VMs | 50 |
| | Processor speed | 500-2000 MIPS |
| | Available memory space in a single VM | 256-2048 Mb |
| | Bandwidth | 500-1000 |
| | Cloudlet Scheduler | Time shared |
| | Number of Processor Elements (PEs) requirement | 1-4 |
| | VM Manager | Xen |
| Task or Cloudlet | Length of task (Executable instruction length in bytes) | 1000-20000 |
| | Total number of tasks | 100-1000 |

### 4.2Comparison

In this section, a comparison is made between three different approaches for load balancing. The three algorithms taken into account are HBB-LB, LBA-HB, and the proposed ILBA-HB. Different metrics can be used to evaluate different techniques; in our work, we use three specific metrics to assess their effectiveness:

1. Response time (RT): The time between a request's initial demand and the first response provided is commonly referred to as response time [30].

$$RT_j = FT - EST \qquad (16)$$

where FT denotes the time taken for the Cloudlet to complete andEST denotes the start time for executing the Cloudlet. The response time for a given $VM_j$ is indicated as $RT_j$, while the average response time for n VMs is defined as:

$$RT = \sum_j \frac{RT_j}{n} \qquad (17)$$

Figures 11 and 12 are box plots representing the distribution of results for RT. The statistical values for these distributions are given in Table 5, which presents a comparison of the response times for ILBA-HB, LBA-HB, and HBB-LB. From

the results, it is clear that the proposed ILBA-HB algorithm improves the response time by 1.93% for the LBA-HB algorithm and 62.71% for the HBB-LB algorithm when the DI value is at its maximum, by 4.38% for the LBA-HB algorithm and 78.05% for the HBB-LB algorithm when the DI value is at its minimum, and by 3.90% for the LBA-HB algorithm and 52.44% for the HBB-LB algorithm when the DI value is average.

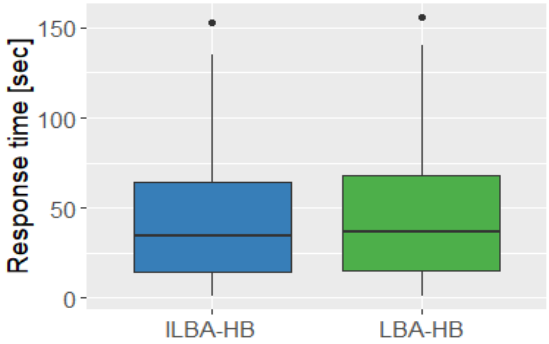


**Figure 11:**RT dispersion for ILBA-HB and LBA-HB

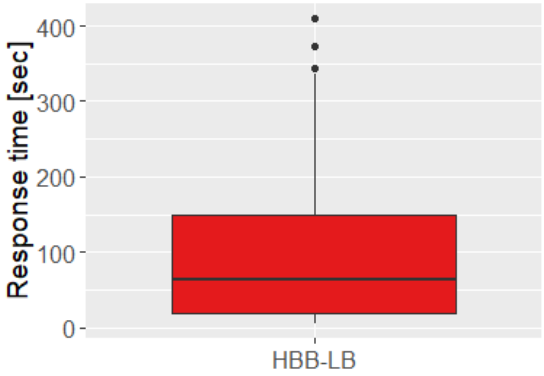


**Figure 12:** RT dispersion for HBB-LB

**Table 5:** Statistics for RT

| Stat | HBB-LB | LBA-HB | ILBA-HB |
|---|---|---|---|
| Min. | 6.36 | 1.46 | 1.396 |
| 1st Qu. | 19.08 | 14.96 | 14.432 |
| Median | 64.28 | 36.70 | 34.922 |
| Mean | 97.40 | 46.32 | 44.513 |
| 3rd Qu. | 148.78 | 68.17 | 64.663 |
| Max. | 409.40 | 155.65 | 152.632 |

2. Makespan is the maximum time it takes to complete a task [10]. M stands for Makespan and is calculated as follows:

$$M = max|CT_{ij}|; \quad i \in T \qquad (18)$$

Figures 13 and 14 indicate the distribution of results for Makespan and Table 6 presents the statistics behind these distributions. Table 6 shows that the proposed algorithm's Makespan was the best performing among cloud computed algorithms in a heterogeneous setting, with the lowest minimum Makespan, maximum Makespan, and average Makespan. ILBA-HB improves Makespan by 3.70% for the LBA-HB algorithm and 84.83% for the HBB-LB algorithm when the DI value is at its maximum, by 33.33% for the LBA-HB algorithm and 91.66% for the HBB-LB algorithm when the DI value is at its minimum, and by 3.55% for the LBA-HB algorithm and 78.30% for the HBB-LB algorithm when the DI value is average.

3. The degree of imbalance (DI) is a metric that calculates the imbalance among VMs, which is defined as follows [8]:

$$T_i = \frac{total\_tasklength}{pe\_num_j \times pe\_mips_j} \qquad (19)$$

where total_tasklength is the total length of the tasks that are submitted to the VMj, $pe\_num_j$ is the number of processors of VMj, and $pe\_mips_j$ is the MIPS of each processor of VMj.

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \qquad (20)$$

Where $T_{max}$ and $T_{min}$ are the maximum and minimum Ti for all VMs, and $T_{avg}$ is the average Ti of the VMs [12].

The distribution and statistics for this measure are shown in Figures 15 and 16 and Table 7.The results of the ILBA-HB algorithm for DI demonstrate that it did not improve the DI when compared to LBA-HB. For the first quartile of DI values and the minimum DI values, the proposed algorithm improved the degree of imbalance slightly over LBA-HB. In contrast, the LBA-HB algorithm outperformed ILBA-HB in its degree of imbalance for the maximum and average values and the third quartile. The performance of the two algorithms for the DI was close, to a large extent, as the two algorithms avoid task migration to keep the system's load balanced.

When comparing the proposed algorithm's output to that of the HBB-LB algorithm in terms of degrees of imbalance, the ILBA-HB algorithm outperformed the HBB-LB algorithm because its minimum DI, maximum DI, and average DI were the lowest in a heterogeneous setting . Table 11 demonstrates how the proposed algorithm improved the degree of imbalance by a significant percentage; the ratio of the ILBA-HB algorithm's improvement to that of the HBB-LB algorithm was 45.34% at the maximum DI, 49.29% at the average, and 50.28% at the minimum.

The proposed algorithm is superior to HBB-LB for the DI as the HBB-LB algorithm migrates tasks between VMs and this causes migration costs. In contrast, the proposed algorithm avoids task migration; we assumed that avoiding task migration reduced the degree of imbalance and this is what the results have proven.

The use of fuzzy logic with the proposed algorithm improves its response time and Makespan compared to those of LBA-HB and HBB-LB, thus improving overall system performance. The ILBA-HB algorithm generates promising results, even if the improvement is negligible.

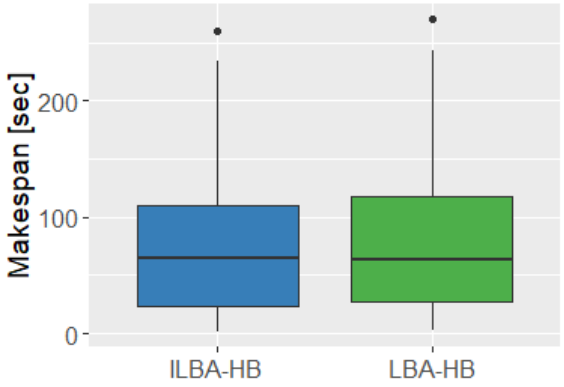


**Figure 13:** Makespan dispersion for ILBA-HB and LBA-HB

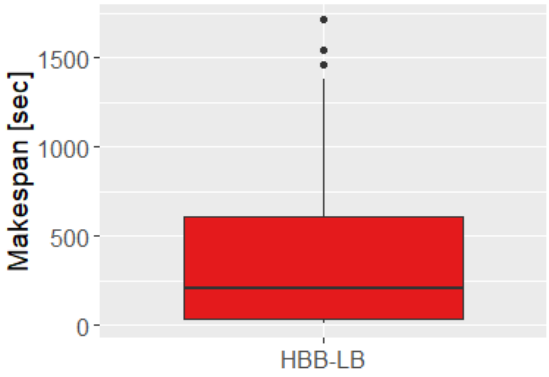


**Figure 14:**Makespan dispersion for HBB-LB

**Table 6:** Statistics for Makespan

| Stat | HBB-LB | LBA-HB | ILBA-HB |
|---|---|---|---|
| Min. | 12.0 | 3.00 | 2.00 |
| 1st Qu. | 36.0 | 27.00 | 24.00 |
| Median | 202.0 | 64.00 | 64.72 |
| Mean | 372.6 | 80.85 | 77.98 |
| 3rd Qu. | 604.1 | 117.74 | 109.75 |
| Max. | 1714.5 | 269.99 | 259.99 |

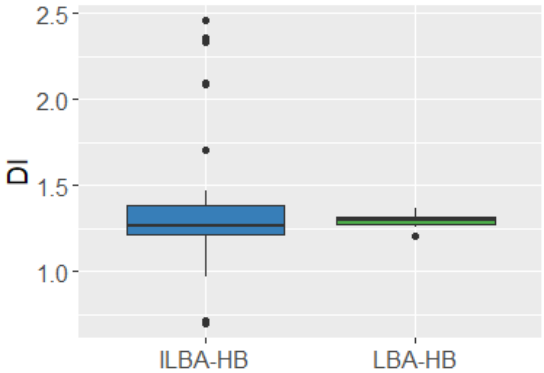


**Figure 15:** DI dispersion for ILBA-HB and LBA-HB

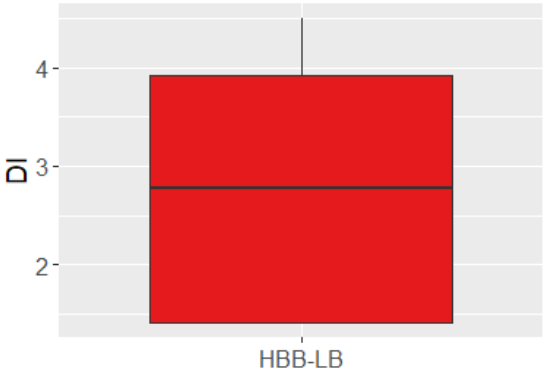


**Figure 16:**DI dispersion for HBB-LB

**Table 7:** Statistics for DI

| Stat | HBB-LB | LBA-HB | ILBA-HB |
|---|---|---|---|
| Min. | 1.414 | 1.205 | 0.703 |
| 1st Qu. | 1.415 | 1.272 | 1.217 |
| Median | 2.772 | 1.301 | 1.265 |
| Mean | 2.700 | 1.297 | 1.352 |
| 3rd Qu. | 3.921 | 1.314 | 1.387 |
| Max. | 4.497 | 1.370 | 2.458 |

The second part of these results discusses the relationship between the VM's power consumption factor and QoS factors, for example cost and processing time. These two variables were the only ones considered in this study. Table 8 presents the power consumed per VM versus the cost factor and processing time factor. The power consumed by the VM is calculated according to Eq. 13 . To illustrate the two variables' relationship, we adopt Spearman's Rank Correlation Coefficient. The Spearman's Rank Correlation Coefficient is a statistical test for determining the degree of correlation between two variables, if any [31]. Meanwhile, a scatter graph for the two variables could indicate whether there is a degree of correlation, with Spearman's Rank providing a numerical value for the degree of correlation, or indeed the degree of non-correlation [31].

Table 9 presents the Spearman's rank correlation coefficient for power consumption, PT, as well as cost. Through this test, it may be concluded that there is a very strong positive correlation between power consumption and cost, because the Rs value is +0.9547, critical probability (p) value is 0.001, while the statistical significance level is 99.9%. The null hypothesis has a 0.1% chance of being correct (p=0.001). Resultantly, we must accept the alternative hypothesis, namely that a very strong positive correlation exists between power consumption and cost, with the null hypothesis rejected (that no correlation exists).

From the results presented in table 9 it may be concluded that power consumption and PT have a very weak positive

correlation, because the Rs value is 0.1589, critical p value is 0.50, while the statistical significance level is 50%. The null hypothesis has a 50% chance of being correct (p=0.50). Consequently, we must accept the null hypothesis that no correlation exists.

**Table 8:** Power consumption, cost, and PT for VMs

| VM _id | power consumption | Cost | PT |
|---|---|---|---|
| 0 | 1.900 | 0.600 | 80.000 |
| 1 | 49.400 | 7.800 | 260.000 |
| 2 | 85.387 | 8.100 | 120.000 |
| 3 | 136.350 | 8.100 | 67.500 |
| 4 | 0.950 | 0.300 | 40.000 |
| 5 | 47.500 | 7.500 | 250.000 |
| 6 | 85.387 | 8.100 | 120.000 |
| 7 | 136.350 | 8.100 | 67.500 |
| 8 | 1.900 | 0.600 | 80.000 |
| 9 | 47.500 | 7.500 | 250.000 |
| 10 | 85.387 | 8.100 | 120.000 |
| 11 | 136.350 | 8.100 | 67.500 |
| 12 | 0.950 | 0.300 | 40.000 |
| 13 | 47.500 | 7.500 | 250.000 |
| 14 | 85.387 | 8.100 | 120.000 |
| 15 | 136.350 | 8.100 | 67.500 |
| 17 | 47.500 | 7.500 | 250.000 |
| 20 | 2.850 | 0.900 | 120.000 |
| 18 | 85.387 | 8.100 | 120.000 |
| 19 | 136.350 | 8.100 | 67.500 |
| 21 | 47.500 | 7.500 | 250.000 |
| 22 | 85.387 | 8.100 | 120.000 |
| 23 | 136.350 | 8.100 | 67.500 |
| 24 | 0.950 | 0.300 | 40.000 |
| 25 | 47.500 | 7.500 | 250.000 |
| 29 | 47.500 | 7.500 | 250.000 |
| 26 | 123.337 | 11.700 | 173.333 |
| 27 | 136.350 | 8.100 | 67.500 |
| 30 | 85.387 | 8.100 | 120.000 |
| 31 | 136.350 | 8.100 | 67.500 |
| 32 | 0.950 | 0.300 | 40.000 |
| 33 | 47.500 | 7.500 | 250.000 |
| 34 | 85.387 | 8.100 | 120.000 |
| 35 | 136.350 | 8.100 | 67.500 |
| 37 | 47.500 | 7.500 | 250.000 |
| 38 | 85.387 | 8.100 | 120.000 |
| 39 | 136.350 | 8.100 | 67.500 |
| 41 | 47.500 | 7.500 | 250.000 |
| 42 | 85.387 | 8.100 | 120.000 |
| 44 | 0.950 | 0.300 | 40.000 |
| 45 | 47.500 | 7.500 | 250.000 |
| 43 | 136.350 | 8.100 | 67.500 |
| 46 | 85.387 | 8.100 | 120.000 |
| 47 | 141.400 | 8.400 | 70.000 |
| 49 | 47.500 | 7.500 | 250.000 |

**Table 9:** Spearman's rank correlation coefficient

| Variables | Rs value | P value |
|---|---|---|
| **PC vs Cost** | 0.9547 | 0.001 |
| **PC vs PT** | 0.1589 | 0.50 |

## 5. CONCLUSION

Cloud computing may be considered as an innovative paradigm providing users with a wide range of resources,which are able toassist with the simultaneous execution of applications or tasks. The numerous applications, as well asconstantly altering user demands, create problems of load balancing—including under-loading and over-loading—for the virtual machines in the cloud data center,resulting in the system's performance being negatively affected.A hybrid algorithm may be used to apply fuzzy logic for implementing a dynamic load balancing algorithm,which can deal with previous algorithms' ambiguity and inconsistency. In order to select the appropriate VM for handling the task, the proposed algorithm takes into account the QoS factors such as cost and processing time, alongside the power consumption factor. The simulation results evidence that the proposed algorithm enhances the mean response time, makespan, as well as degree of imbalance over LBA_HB and HBB-LB. Additionally, it clarified that energy consumption and cost have a very strong positive correlation,while the correlation between energy consumption and processing time is very weak. However, future researchcould consider the new meta-heuristic nature-inspired techniques for ensuring load-balancing with regards to energy consumption and QoS. Moreover, it is feasible to apply the proposed algorithm in the real time environment as a means of comparing the results.

## REFERENCES


1. 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.
2. P. T. Endo, M. Rodrigues, G. E. Gonçalves, J. **availability in clouds: systematic review and research challenges**," *J. Cloud Comput.*, vol. 5, no. 1, 2016, doi: 10.1186/s13677-016-0066-8.
3. 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.
4. 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.
5. D. Satria, D. Park, and M. Jo, "**Recovery for overloaded mobile edge computing**," *Futur. Gener. Comput. Syst.*, vol. 70, pp. 138–147, 2017, doi: 10.1016/j.future.2016.06.024.

6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. S. Mustafa, B. Nazir, A. Hayat, A. U. R. Khan, and S. A. Madani, "**Resource management in cloud computing: Taxonomy, prospects, and challenges**," *Comput. Electr. Eng.*, vol. 47, pp. 186–203, 2015, doi: 10.1016/j.compeleceng.2015.07.021.
16. 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.
17. M. Tan, C. Chi, J. Zhang, S. Zhao, G. Li, and S. Lü, "**An energy-aware virtual machine placement algorithm in cloud data center**," *ACM Int. Conf. Proceeding Ser.*, vol. Part F1318, no. C, pp. 719–723, 2017, doi: 10.1145/3144789.3144792.
18. T. Deepika and P. Prakash, "**Power consumption prediction in cloud data center using machine learning**," *Int. J. Electr. Comput. Eng.*, vol. 10, no. 2, pp. 1524–1532, 2020, doi: 10.11591/ijece.v10i2.pp1524-1532.
19. 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.
20. M. G. K. Geetha Megharaj, **Metaheuristic-Based Virtual Machine Task Migration Technique for Load Balancing in the Cloud**, vol. 771. Springer Singapore, 2019.
21. 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.
22. 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.
23. 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.
24. 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.
25. 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.
26. 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.
27. A. P. S.Yakhchi,S.M.Ghafari, M.YakhchiM.Fazeli, "**ICA-MMT: A load balancing method in cloud computing environment**.," 2015, [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7210303/.
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. N. Mansouri, B. Mohammad Hasani Zade, and M. M. Javidi, "**Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory**," *Comput. Ind. Eng.*, vol. 130, no. March, pp. 597–633, 2019, doi: 10.1016/j.cie.2019.03.006.
30. 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.
31. H. Akoglu, "**User's guide to correlation coefficients**," *Turkish J. Emerg. Med.*, vol. 18, no. 3, pp. 91–93, 2018, doi: 10.1016/j.tjem.2018.08.001.