Daisy Sharmah
Kanak Chandra Bora
Junumoni Khakhlari

MODIFIED HONEY BEE ALGORITHM WITH RANDOM SELECTION OF VIRTUAL MACHINES FOR DYNAMIC LOAD BALANCING

Abstract *Cloud workloads can overwhelm load balancers, leading to inefficiencies and performance issues. To address these challenges, the honey bee load-balancing algorithm is highly effective in enhancing cloud-resource allocation. Inspired by the foraging behavior of honey bees, this algorithm offers a dynamic approach to resource distribution that adapts to changing workloads in real time. This paper delves into the key features and advantages of honey bee load-balancing, focusing on its dynamic resource allocation, overall response time, and data center processing time. Through a comparative study of existing methodologies, we proposed a modified honey bee load-balancing algorithm that incorporated the random selections of virtual machines. Utilizing the CloudAnalyst tool for simulation, we compares traditional and proposed honey bee load-balancing algorithms to evaluate overall response times and data center processing times using user bases, data centers, virtual machine load balancers, time, service broker policies, and regions. The proposed algorithm demonstrates superior performance in these parameters as compared to the traditional approach when using the same metric values for both algorithms.*

Keywords dynamic load balancing, honey bee foraging, virtual machine, response time, data center

Citation Computer Science 26(3) 2025: 73–91

1. Introduction

In today's world of cloud computing (CC), computing resources have become a concern for both service providers and users. The various workloads in CC need high availability and cost effectiveness as well as different challenges to overcome in load-balancing (LB) techniques. LB is used to reduce the makespan and response times of the CC infrastructure [5]. To address these challenges, an approach that is known as honey bee-based LB has been discussed as a promising solution. By taking inspiration from the identical behaviors of honey bees, this innovative approach seeks to revolutionize how CC resources are managed. Honey bees can optimize resource allocation while changing environmental conditions by adopting various CC resources. In this paper, we focused on exploring the algorithm called honey bee-based load-balancing in cloud computing based on the random selection of virtual machines. We will dig into the core concepts, mechanisms, and advantages of this approach and how it optimizes resource allocation, enhances fault tolerance, reduces operational costs, and improves the overall performance and efficiency of cloud services by examining the innovative approaches of this technique and its potential to address the challenges in cloud-resource allocation [12].

1.1. Objective

The main objective of this paper was to propose a modified honey bee algorithm that improves performance by selecting randomized virtual machines for the utilization of dynamic load balancing (LB). The proposed algorithm was targeted to minimize the average response time and average data center processing time. The proposed algorithm was compared with the traditional honey bee algorithm using a simulation tool to analyze the result of the overall response time and data center processing time.

1.2. Load-balancing mechanism

This was the chance to express the work with the considerations that were collected in the above steps by taking on any of the reasonable methodologies. LB plays an important role in maintaining activities in a CC environment; it helps to minimize the response time in order to avoid system overload, maximize the throughput, and obtain optimal resource utilization [26]. The purpose of LB is to avoid the overloading and idleness of the nodes in a cloud system. In CC platforms such as Windows Azure Platform, Amazon S3, etc., as well as its use in artificial intelligence [27], this will enhance the development of many web searches with distinctive features from cloud service providers. LB algorithms are necessary, as they provide continuous services to users without service breaks.

1.3. Dynamic load-balancing algorithms

In a dynamic algorithm, the server with the least load in an entire network or system is identified and prioritized for load balancing. Real-time communication with the network is required for this, which can increase system traffic. Here, the current

state of the system is used to make decisions to manage the load [19]. Dynamic algorithms make load-transfer decisions based on the actual current state of the system. Since dynamic load-balancing decisions are based on the current state of the system, processes can be transferred from an overutilized machine to an underutilized one in real time [24].

1.4. Use of honey bee algorithm in load balancing

It is a nature-inspired dynamic load-balancing technique that helps to achieve load balancing across the heterogeneous virtual machines of a cloud-computing environment through local server action and maximizes the throughput. The current workload of the VM is evaluated to determine whether the VM is overloaded, underloaded, or balanced. VMs are grouped based on their current loads; the priority of the tasks is considered after they are removed from the overloaded VMs and are waiting for assignment to other VMs. The previously removed tasks help to identify a lightly loaded VM and are referred to as scout bees in the next step. The honey bee behavior-inspired load-balancing technique reduces the response times of VMs and also reduces the waiting times of the tasks [11].

Lemmens et al. [14], entitled "Bee Behavior in Multi-Agent Systems," mentioned that they presented a new non-pheromone-based algorithm that was inspired by the behavior of bees. The algorithm combined both recruitment and navigation strategies. They investigated whether this new algorithm outperformed pheromone-based algorithms (inspired by the behavior of ants) in the task of foraging. They said that, for navigation, they used a strategy named "path integration (PI)." This strategy was based on the path integration in bees, which enables them to continuously calculate their current locations from their past trajectories. As a result, they can return to their starting points by taking direct routes instead of retracing their outbound paths. Their algorithm consisted of two strategies; these are mentioned below [28].

First, a recruitment strategy is used to distribute knowledge to the other members of the colony. More precisely, the agents can communicate the distance and direction to a destination by 'dancing' inside the hive (analogous to how bees perform their 'dance' inside the hive). Second, a navigation strategy is used to efficiently navigate an unknown world.

The honey bee algorithm is a nature-inspired decentralized load-balancing strategy that assists with accomplishing load balancing across the heterogeneous virtual machines of a distributed computing environment through nearby server activity and expands the throughput [23]. The VM's current workload is assessed to determine whether the VM is overloaded, underloaded, or balanced. VMs are grouped based on their current loads. The priority of the task is taken into consideration after being removed from an overloaded VM, which is waiting for the VM [8]. The earlier removed tasks help find lightly loaded VM; these tasks are known as scout bees in the next step. The honey bee behavior-inspired LB technique reduces the response times of VMs and also reduces the waiting times of the tasks [11].

2. Related works

Singh et al. [26] compared the results that were obtained from different LB algorithms; the response times of the throttled VMs were as good when matched to the RR and ESCE LB algorithms. Also, the processing times that were taken by the DC were good for both of these algorithms.

Lemmens et al. [14] presented a new non-pheromone-based algorithm that was inspired by the behavior of bees. They investigated whether this new algorithm outperformed pheromone-based algorithms (inspired by the behavior of ants) in the task of foraging.

Wickremasinghe et al. [29] mentioned that the simulation settings and experimental results were also in this paper, with variations in the average response times. By bringing the service closer to users, their results showed that it improved the quality of the service based on the response time.

Inspired by LB of tasks in the CC environment, Venkata et al. [5] mentioned that scheduling tasks in CC was an NP-hard optimization problem. The LB of non-preplanned autonomous assignments on virtual machines (VMs) is a significant part of task scheduling for clouds; in this paper, they proposed an algorithm that was named HBBLB (which aimed to achieve a well-balanced load across virtual machines to maximize the throughput).

Kumar S. et al. [13] conducted a comparison of various dynamic LB algorithms based on seven key metrics: fault tolerance, throughput, performance, resource utilization, response time, migration time, and overhead. The analysis revealed differences across the algorithms and highlighted those areas where improvements could be made. Specifically, enhancing the fault tolerance, increasing the throughput, and optimizing the resource utilization were identified as areas for potential improvements. Additionally, reducing the response and migration times while minimizing the overhead that was associated with load balancing were seen as critical factors for improving the overall performance.

Bhavya et al. [3] performed a honey bee foraging algorithm based on a throughput that was calculated for each virtual machine. Their goal was to find the appropriate VM with the help of the throughput that was calculated earlier and allocate the load (which led to a more efficient LB).

Hashem et al. [9] proposed an LB algorithm based on honey bee behavior; their main goal was to distribute the workload of multiple network links in such a way that avoided the underutilization and overutilization of the resources. The proposed algorithm calculation was mimicked utilizing CloudSim. The proposed calculation was contrasted with both customary and SI-based LB calculations, RR, modified throttled, ant colony, and HBAs.

Priyanka et al. [17] mentioned that, in a cloud environment, a proper LB avoids fail-over and bottlenecks; this in turn improved the flexibility and scalability of the resources, reduced the over-provisioning of the VM allocation, and minimized

the resource utilization. This paper presented a detailed survey of LB algorithms, along with the metrics. In the future, however, research work will be carried out on the conservation of energy on the cloud as well as workload optimization by providing the efficient computation and effective processing of resources.

Gupta et al. [7] considered two data centers and four VMs for their research and for presenting the resulting paradigm. The overall response time and the response time by various regions were illustrated by the authors in this paper; the cost efficiency of the proposed system was also mentioned in a graphical representation. Mishra et al. [15] described various LB techniques in homogeneous and heterogeneous CC environments. The calculation of the make-span and energy consumption of the system was explained in detail in this paper.

Afzal et al. [1] mentioned that their paper presented a detailed encyclopedic review of LB techniques. The study was conducted with a general research strategy where the load-unbalancing problem was undertaken, and the researchers identified the methods, theories, algorithms, approaches, and paradigms that were used in it.

Shukla et al. [25] presented several static and dynamic LB approaches under different categories. However, the entire algorithm mainly discussed overloaded and underloaded processes with threshold values. The author explained the merits and demerits of each algorithm.

Kumar et al. [13] provided a comparative analysis of studied articles based on various metrics that they used in their work in the CC environment. The researchers were focused on minimizing the response time to maintain the SLA and increasing the response time.

Ullah et al. [28] elaborated on the use of artificial bee colony (ABC) in LB. The author mentioned that the use of ABC could improve the performance of VM scheduling.

Ebadifard et al. [6] stated that using an appropriate LB method could reduce the response time and increase the resource utilization; the proposed method reduced the makespan, increased the degree of the LB, and improved the system's reliability. They compared the proposed algorithm with other task-scheduling algorithms, such as the honey bee LB and dynamic scheduling without LB.

Kalaivani et al. [10] mentioned that feature selection (FS) played an essential role in creating machine-learning models. In this research work, they proposed a technique for selecting container functions for IDS. The proposed hybrid MBC-BFO algorithm was analyzed with three different classifications.

Pushpavati et al. [18] mentioned that the paired tree algorithm had 25 cloudlets, giving task migrations of 7 and 91.32% of the makespan.

Parida et al. [16] mentioned LB as the most dynamic problem in the CC environment; they proposed the Salp swarm-optimization technique as a meta-heuristic algorithm for solving this problem. The authors also proposed the binary version of the Salp swarm-optimization algorithm to solve the problems in binary task assignments.

Aghdai et al. [2] described a scalable LB architecture (mentioned as 'Spotlight') to maintain the mapping between the networks of DC. They used PCC to improve the distribution of new services among the LB. They also stated that the PCC in the LB had mainly two aspects: maintaining the PCC, and flow dispatching.

Shahid et al. [21] presented a performance evaluation of the existing load-balancing algorithms, which were PSO, RR, ESCE, and TLB. This study gave us a detailed evaluation performance of various load-balancing algorithms that were performed by the cloud-analyst platform.

Zhao et al. [30] proposed an improved artificial bee colony (IABC) optimization algorithm using the flying ad hoc network (FANET) to improve the convergence rate and exploitation abilities. They used a super mining node for block verification and block processing.

Rani et al. [20] mentioned the modified honey bee behavior-inspired load-balancing (HBBLB) algorithm using the modified blowfish technique and experimented so the migration tasks were reduced to 30, 25, and 20% based on various criteria.

Brahmam et al. [4] stated that VM migration with deep-learning optimization reduced the makespan by 4.5% and increased task diversity by 3.9%.

Table 1 depicts the state of the art of the related works in chronological order.

Table 1
Meta-analysis of literature review (yearwise)

Reference	Algorithm used	Advantages	Limitations
[1]	HBFA	Reduces execution time	degraded VM migration
[14]	Bee-inspired algorithm	Less computation time	Less adaptive
[5]	HBBLB	Reduces minimum waiting time	QoS Factor considered
[3]	MHBFA	Minimizes overall response time	–
[9]	HBFA	Reduces execution time	Degraded VM migration
[28]	ABC	Improves VM Scheduling performance	VM replacement not used
[6]	HBLB	Reduces waiting time	No cost-reducing factors
[10]	MBC-BFO	Accuracy rate is low	–
[18]	EBC	Improved task migration	Fewer QoS parameters
[30]	IABC	Enhances exploitation abilities	–

Table 1 cont.

Reference	Algorithm used	Advantages	Limitations
[20]	HBBLB	Migration tasks reduced	LB Security not checked
[4]	VM Migration	Task diversity increased	Not energy-efficient

3. Methodology

In the proposed model, we modified the honey bee foraging algorithm to balance the load of the CC paradigm by equally distributing the tasks among the virtual machine (VM) by selecting random VMs based on two criteria. The calculations of the throughput and the threshold value are stated below.

3.1. Calculation of throughput

The throughput is the unit of the information that a system can process in a given amount of time; it is the calculation of several processes by the total scheduled time.

In calculation:

$$T_p = M_t / P^t$$

Here, T_p is the throughput of a system.

M_t is the maximum completion time.

P^t marks the initiation (or commencement) of the first process.

In the concept of LB, the throughput can be calculated for a virtual machine (as illustrated in the following example):

Task T1 takes 3 seconds for execution, task T2 takes 5 seconds for execution, and task T3 takes 10 seconds for execution; then, the throughput is calculated as follows:

$T_p = (3 + 5 + 10)/3 = 18/3 = 6$ seconds.

The threshold value in LB is the value that initiates the LB action when it is reached; it is considered to be a percentage of the maximum capacity based on the operational needs and tolerance for the performance degradation. The calculation of the threshold value is as follows:

Threshold value capacity of resources X utilization required in percentage.

Let us consider an example of CPU capacity as 20 Cores in 55.0 GHz each and the desired utilization threshold as 90%; then, the threshold value (depicted as T.V.) is calculated as follows:

T.V. = Total CPU usage X 90% = (20 Cores $\cdot$ 5.0 GHz) $\cdot$ 0.9 = 90 GHz.

3.2. Proposed algorithm

Step 1: Start

Step 2: Set the Number of Tasks, Number of Virtual Machine

Step 3: Set all the load of VM to 0

Step 4: A server sends a request

Step 5: The load balancer selects a VM on a random basis

Step 6: If the load of the current VM <=Threshold value

Step 6.1: If yes, check the current VM with high throughput

Step 6.1.1: If yes, Run the current task on this VM

Step 6.1.2: If No, Go to Step 5

End If

Step 6.2: If No, Go to Step 5

End If

Step 7: If all the Tasks are not completed by VMs

Step 7.1: If yes, Go to Step 4

End If

Step 8: Stop

3.3. Workflow of proposed algorithm

The proposed algorithm is stated, and the workflow is shown in Figure 1 above. Initially, the VM is set to 0 for all of the VMs. As the server sends a request, the load balancer selects a VM on a random basis. The load of the selected VM is compared with the threshold value; if the current VM value is less than or equal to the threshold value (which was set earlier), then it will check the throughput of the current VM. If the throughput of the current VM is high, then the task is assigned to the current VM; otherwise, the load balancer selects another VM randomly once again. If all of the tasks are completed by the VMs, then the process stops; otherwise, it waits for the server to send the next request.

In the context of the proposed load-balancing technique, the tasks that are removed from the overloaded virtual machines (VMs) are prioritized based on the criteria of the high throughput and low threshold values; this prioritization ensures that the tasks are reassigned to those VMs that are capable of processing them efficiently, thus minimizing any delays and optimizing the resource utilization. If no suitable VM is identified based on these criteria, the selection process continues randomly until an appropriate VM is found. The "scout bees" in the algorithm play crucial roles in identifying lightly loaded VMs for task reassignment. Drawing inspiration from the behavior of scout bees in nature (which search for new food sources), these algorithmic counterparts continuously monitor the cloud environment to detect underutilized resources.

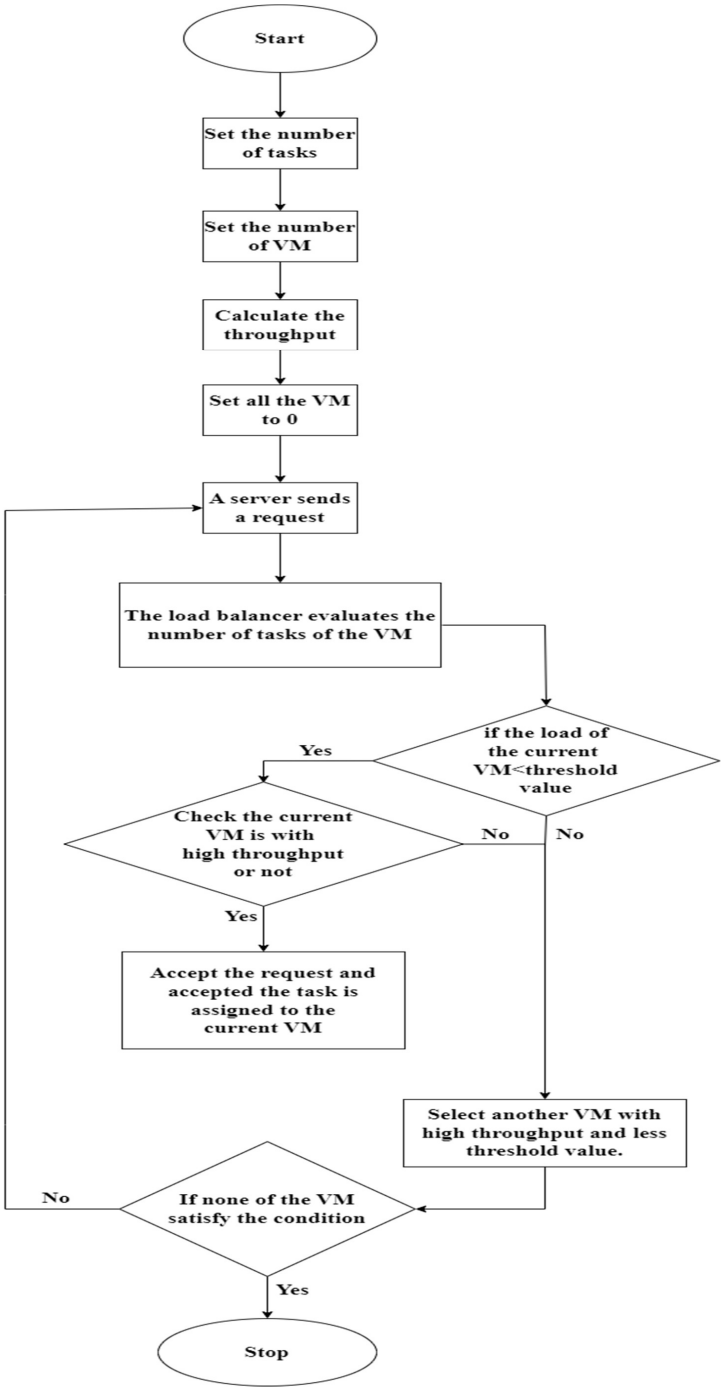

Figure 1. Workflow of proposed algorithm

Scout bees evaluate the workload and resource availability of the VMs and report their findings; this distributed and dynamic scouting process ensures that the load-balancing system can adapt to fluctuating workloads and resource demands in real time, thus enabling the efficient redistribution of the tasks to the lightly loaded VMs.

4. Results and discussion

4.1. CloudAnalyst

CloudAnalyst is a tool that was designed to evaluate various load-balancing algorithms. Its primary purpose is to provide insights into simulation issues, thus allowing users to concentrate on their analyses without getting entangled in the complexities of programming with a simulation toolkit [22]. Additionally, CloudAnalyst enables modelers to run simulations repeatedly and perform multiple simulation experiments swiftly and efficiently with minor parameter adjustments. By separating the simulation setup from the technical execution, CloudAnalyst allows modelers to focus on the parameters for their simulations rather than just the programming details. The cloud-environment simulation is conducted using CloudAnalyst, a tool that was built on the open-source CloudSim library suite [19]. Java is used as the programming language, and Eclipse serves as the development environment.

We used CloudAnalyst to simulate and analyze for a better LB algorithm. We compared our proposed algorithm with the traditional honey bee foraging algorithm (HBFA) and simulated the results of both using CloudAnalyst. For both algorithms, we adapted the parameters that are listed in Table 2.

Table 2
List of parameters and their values

Slno	Parameter name	Value
1.	USER BASES (UB)	6
2.	DATA CENTER (DC)	6
3.	VM LOAD BALANCER	10 in each DC
4.	TIME	60 mins
5.	SERVICE BROKER POLICY	Reconfigure Dynamically Load
6.	REGIONS	6(0-5)

4.2. Experimental tests

We considered two matrices: i – Overall response time in MS; and ii – DCPT. The simulations that were performed in CloudAnalyst are shown in Figures 2 and 3.

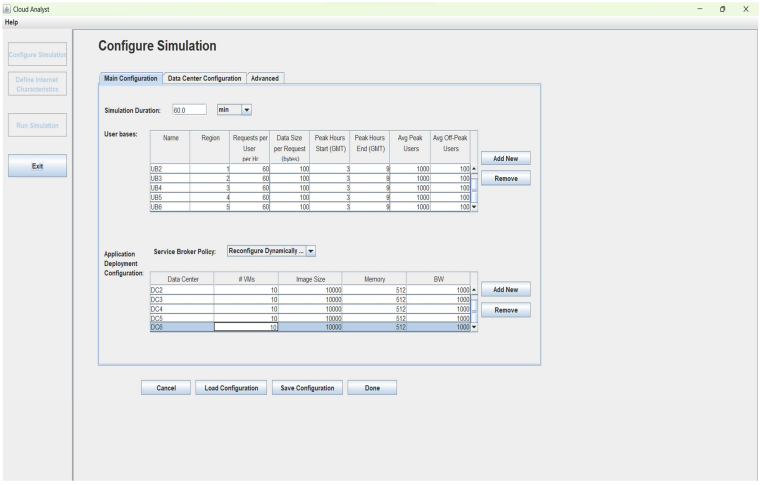

Figure 2. Configuring CloudAnalyst

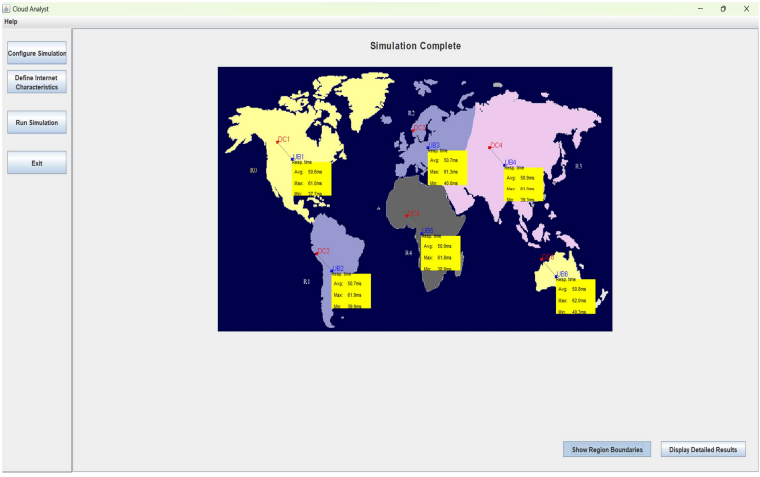

Figure 3. After completing simulation

The experimental results show the comparison work between the traditional honey bee foraging algorithm (HBFA) and the proposed algorithm in Tables 3 through 8.

Table 3

Overall response time (RT) summary of traditional honey bee foraging algorithm (HBFA)

Summary	Average RT [MS]	Minimum RT [MS]	Maximum RT [MS]
Overall response time	52.65	37.65	70.02
Data center processing time	3.03	0.02	15.30

Table 4
Response times (RTs) by region of traditional HBFA

Userbase	Average RT [MS]	Minimum RT [MS]	Maximum RT [MS]
UB1	50.97	37.65	61.70
UB2	53.09	39.89	70.02
UB3	53.24	40.13	68.05
UB4	52.95	39.36	69.36
UB5	52.83	38.93	69.63
UB6	52.83	40.26	67.60

Table 5
Data center (DC) request servicing times of traditional HBFA

Data center	Average RT [MS]	Minimum RT [MS]	Maximum RT [MS]
DC1	1.26	0.02	5.57
DC2	3.61	0.09	14.31
DC3	3.76	0.04	15.30
DC4	3.18	0.06	12.08
DC5	3.18	0.04	13.09
DC6	3.22	0.04	14.18

Table 6
Overall response time (RT) summary of proposed algorithm

Summary	Average RT [MS]	Minimum RT [MS]	Maximum RT[MS]
Overall response time	50.57	37.71	62.03
Data center processing time	1.13	0.03	3.46

Table 7
Response times (RTs) by region of proposed algorithm

Userbase	Average RT [MS]	Minimum RT [MS]	Maximum RT [MS]
UB1	50.56	37.71	60.98
UB2	50.72	39.89	61.90
UB3	50.71	40.01	61.35
UB4	50.89	38.93	61.83
UB5	50.86	38.93	61.83
UB6	52.79	40.26	62.03

Table 8
Data center (DC) request servicing times of proposed algorithm

Data center	Average RT [MS]	Minimum RT [MS]	Maximum RT [MS]
DC1	0.82	0.03	1.58
DC2	1.23	0.07	2.38
DC3	1.26	0.04	2.43
DC4	1.16	0.06	2.30
DC5	1.19	0.04	2.34
DC6	1.15	0.04	3.46

From Figures 4 and 5, it is clear that our proposed algorithm worked better when compared to the honey bee foraging algorithm in terms of the average response time, maximum response time, average DCPT processing time, and maximum DCPT.

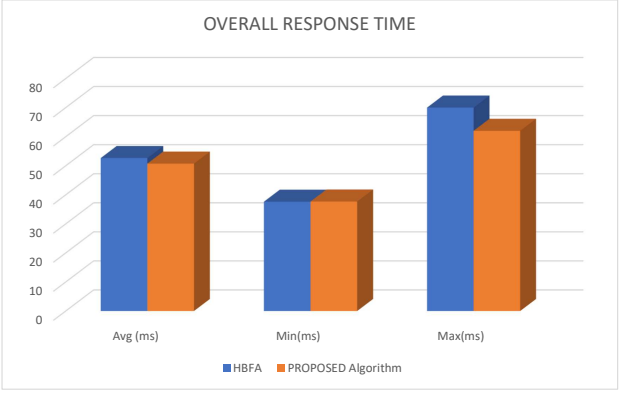

Figure 4. Overall response time comparison

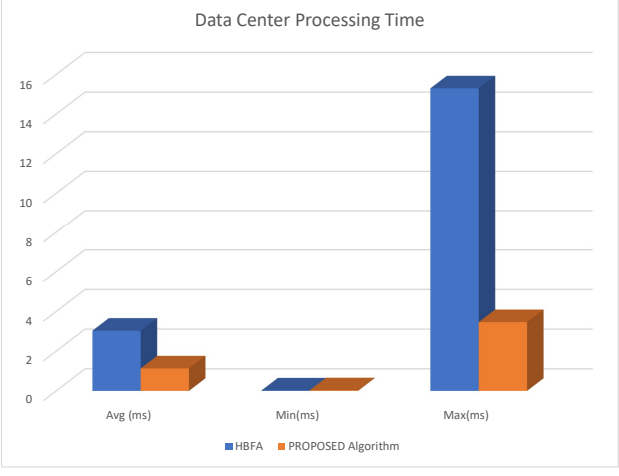

Figure 5. DCPT comparison

Comparisons with the other three algorithms (RR, ESCE, and TLB) along with traditional HBA were made with the proposed algorithm; the convergence graph for the average response time and the data center processing time (DCPT) are shown in Figures 6 and 7, respectively.

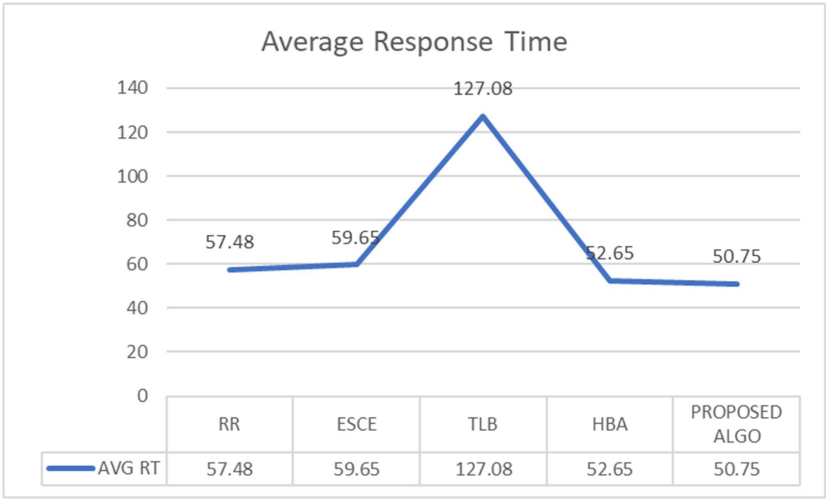

Figure 6. Comparison graph for average response times

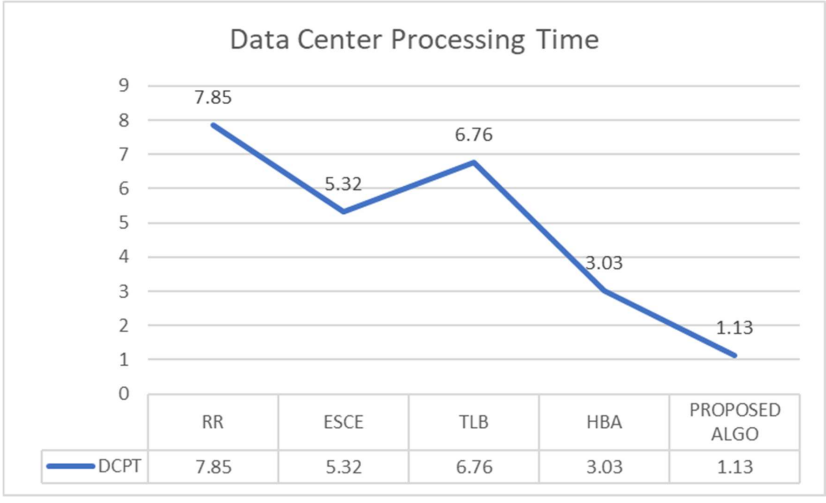

Figure 7. Comparison graph for DCPT

From the comparison graph in Figures 6 and 7, it is clear that our proposed algorithm worked significantly well in average RT and DCPT for the metrics (as was

stated in Table 2). The average RTs of our proposed algorithm were 11% better than RR, 15% better than ESCE, 60% better than TLB, and 3.6% better than traditional HBA. The DCPT performance of the proposed algorithm was 85% better than RR, 78.8% better than ESCE, 83% better than TLB, and 62.7% better than traditional HBA.

5. Conclusion

The application of a honey bee-inspired algorithm for load balancing in cloud computing introduces an innovative approach to addressing the challenges of the efficiently distributions of workloads across cloud resources. The proposed work outlines a modified honey bee-based algorithm for load balancing in cloud environments.

This algorithm selects a virtual machine (VM) at random and evaluates it based on two key criteria: high throughput, and a low threshold value. If the VM meets these criteria, the server's request is assigned to it; otherwise, another VM is randomly selected, and the evaluation process is repeated. This modified algorithm demonstrates advantages over the traditional honey bee foraging algorithm by achieving improved response times and data center processing efficiency. By drawing on nature-inspired principles of distributed, adaptive, and efficient resource allocation, the proposed approach offers a promising and distinctive solution for load balancing in cloud computing.

The proposed algorithm was evaluated under varying workloads in a simulated environment with ten virtual machines. Compared to the traditional honey bee foraging algorithm, the proposed method demonstrated a 3.6% reduction in the average response time (50.75 vs. 52.65 ms) and a 62.7% improvement in the data center processing time (1.13 vs. 3.03 ms).

6. Future scope

The practicality and effectiveness of the proposed algorithm depend on its specific implementation, careful parameter tuning, and extensive testing in real-world cloud environments. We plan to explore this approach to assess its potential benefits and limitations in various cloud-computing scenarios. The dynamic resource allocation to adapt real-time changes using the proposed algorithm is also one of our future scopes.

Conflict of interest

There is no one involved in a conflict of interest in this paper.

List of abbreviations

Definition	Abbreviation
RR	Round Robin
ESCE	Equally Spread Current Execution
VM	Virtual Machine
HBA	Honey Bee Algorithm
TLB	Throttled Load Balancing
HBLB	Modified Honey Load Balancing
HBFA	Honey Bee Foraging Algorithm
MHBFA	Modified Honey Bee Foraging Algorithm
DC	Data Center
CC	Cloud Computing
LB	Load Balancing
MS	Milliseconds
UB	User Bases
RT	Response Time
ORT	Optimized Response Time
DCPT	Data Center Processing Time
PCC	Pre-Connection Consistency
SLA	Service Level Agreement
HBBLB	Honey Bee Behavior-Inspired Load Balancing
MBC-BFO	Hybrid Modified Bee Colony-Bacterial Foraging Optimization
EBC	Enhanced Bee Colony
ABC	Artificial Bee Colony
IABC	Improved Artificial Bee Colony
QoS	Quality of Service

References

[1] Afzal S., Kavitha G.: Load balancing in cloud computing – A hierarchical taxonomical classification, *Journal of Cloud Computing*, vol. 8, 22, 2019. doi: 10.1186/s13677-019-0146-7.

[2] Aghdai A., Chu C.Y., Xu Y., Dai D.H., Xu J., Chao H.J.: Spotlight: Scalable Transport Layer Load Balancing for Data Center Networks, *IEEE Transactions on Cloud Computing*, vol. 10(3), pp. 2131–2145, 2022. doi: 10.1109/TCC.2020.3024834.

[3] Bhavya V.V., Rejina K.P., Mahesh A.S.: An Intensification of Honey Bee Foraging Load Balancing Algorithm in Cloud Computing, *International Journal of Pure and Applied Mathematics*, vol. 114, pp. 127–136, 2017.

[4] Brahmam M.G., Vijay Anand R.: VMMISD: An Efficient Load Balancing Model for Virtual Machine Migrations via Fused Metaheuristics with Iterative Security Measures and Deep Learning Optimizations, *IEEE Access*, vol. 12, pp. 39351–39374, 2024. doi: 10.1109/ACCESS.2024.3373465.

[5] Dhinesh Babu L.D., P. Venkata Krishna: Honey bee behavior inspired load balancing of tasks in cloud computing environments, *Applied Soft Computing*, vol. 13(5), pp. 2292–2303, 2013. doi: 10.1016/j.asoc.2013.01.025.

[6] Ebadifard F., Babamir S.M., Barani S.: A Dynamic Task Scheduling Algorithm Improved by Load Balancing in Cloud Computing. In: *2020 6th International Conference on Web Research (ICWR)*, pp. 177–183, IEEE, Tehran, Iran, 2020. doi: 10.1109/ICWR49608.2020.9122287.

[7] Gupta A.: Load Balancing in Cloud Computing, *International Journal of Distributed and Cloud Computing*, vol. 5(2), pp. 22–28, 2017.

[8] Gupta H., Sahu K.: Honey Bee Behavior Based Load Balancing of Tasks in Cloud Computing, *International Journal of Science and Research*, vol. 3(6), 2012.

[9] Hashem W., Nashaat H., Rizk R.: Honey Bee Based Load Balancing in Cloud Computing, *KSII Transactions on Internet and Information Systems*, vol. 11(12), pp. 5694–5711, 2017. doi: 10.3837/tiis.2017.12.001.

[10] Kalaivani S., Gopinath G.: Modified Bee Colony With Bacterial Foraging Optimization Based Hybrid Feature Selection Technique for Intrusion Detection System Classifier Model, *ICTACT Journal on Soft Computing*, vol. 10(04), pp. 2146–2152, 2020.

[11] Kashyap D., Viradiya J.: A Survey Of Various Load Balancing Algorithms In Cloud Computing, *International Journal of Scientific and Technology Research*, vol. 3(11), pp. 115–119, 2014.

[12] Kitchenham B.: Procedures for Performing Systematic Reviews, Keele University Technical Report TR/SE-0401, NICTA Technical Report 0400011T1, 2004.

[13] Kumar P., Kumar R.: Issues and Challenges of Load Balancing Techniques in Cloud Computing: A Survey, *ACM Computing Surveys*, vol. 51(6), 120, 2019. doi: 10.1145/3281010.

[14] Lemmens N., De Jong S., Tuyls K., Nowä A.: Bee Behaviour in Multi-agent Systems: A Bee Foraging Algorithm. In: K. Tuyls, A. Nowe, Z. Guessoum, D. Kudenko (eds.), *Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning, 5th, 6th, and 7th European Symposium, ALAMAS 2005–2007 on Adaptive and Learning Agents and Multi-Agent Systems, Revised Selected Papers*, vol. 4865, pp. 145–156, Springer, Berlin–Heidelberg, 2008. doi: 10.1007/978-3-540-77949-0_11.

[15] Mishra S.K., Sahoo B., Parida P.P.: Load balancing in cloud computing: A big picture, *Journal of King Saud University – Computer and Information Sciences*, vol. 32(2), pp. 149–158, 2020. doi: 10.1016/j.jksuci.2018.01.003.

[16] Parida B.R., Rath A.K., Mohapatra H.: Binary Self-Adaptive Salp Swarm Optimization-Based Dynamic Load Balancing in Cloud Computing, *International Journal of Information Technology and Web Engineering*, vol. 17(1), pp. 1–25, 2022. doi: 10.4018/IJITWE.295964.

[17] Priyanka M., Sivagami V.M.: A Survey on Load Management Techniques in Cloud Computing, *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 2(2), pp. 1115–1121, 2017. https://ijsrcseit.com/paper/CSEIT1722392.pdf.

[18] Pushpavati S. D Mello D.A.: A tree-based mechanism for the load balancing of virtual machines in cloud environments, *International Journal of Information Technology*, vol. 13, pp. 911–920, 2021. doi: 10.1007/s41870-020-00544-3.

[19] Raghava N.S., Singh D.: Comparative Study on Load Balancing Techniques in Cloud Computing, *Open Journal of Mobile Computing and Cloud Computing*, vol. 1(1), pp. 18–25, 2014.

[20] Rani P., Singh P.N., Verma S., Ali N., Shukla P.K., Alhassan M.: An Implementation of Modified Blowfish Technique with Honey Bee Behavior Optimization for Load Balancing in Cloud System Environment, *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–14, 2022. doi: 10.1155/2022/3365392.

[21] Shahid M.A., Alam M.M., Su'ud M.M.: Performance Evaluation of Load-Balancing Algorithms with Different Service Broker Policies for Cloud Computing, *Applied Sciences*, vol. 13(3), 1586, 2023. doi: 10.3390/app13031586.

[22] Shakir M.S., Razzaque A.: Performance Comparison of Load Balancing Algorithms using Cloud Analyst in Cloud Computing. In: *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, pp. 509–513, IEEE, Newyork, USA, 2017. doi: 10.1109/UEMCON.2017.8249108.

[23] Sharmah D., Bora K.C.: A Survey on Dynamic Load Balancing Techniques in Cloud Computing. In: M. Gabbouj, S.S. Pandey, H.K. Garg, R. Hazra (eds.), *Emerging Electronics and Automation. Select Proceedings of E2A 2022,* Lecture Notes in Electrical Engineering, vol. 1088, pp. 273–282, Springer Nature Singapore, Singapore, 2024. doi: 10.1007/978-981-99-6855-8_21.

[24] Sharmah D., Bora K.C., Noorain M., Karmacharya Y.: Utilizing dynamic load balancing to improve private cloud paradigm, *International Journal of Information Technology*, vol. 16(6), pp. 3465–3474, 2024. doi: 10.1007/s41870-024-01888-w.

[25] Shukla S., Suryavanshi R.S.: Survey on Load Balancing Techniques, *International Conference of Emerging Trends in Technology and Application*, 2019.

[26] Singh S.P., Sharma A., Kumar R.: Analysis of Load Balancing Algorithms using Cloud Analyst, *International Journal of Grid and Distributed Computing*, vol. 9(9), pp. 11–24, 2016. doi: 10.14257/ijgdc.2016.9.9.02.

[27] Tadapaneni N.R.: Artificial Intelligence in Software Engineering, *Social Science Research Network*, vol. 3, 2017.

[28] Ullah A., Nawi N.M., Uddin J., Baseer S., Rashed A.H.: Artificial Bee Colony algorithm used for load balancing in cloud computing: Review, *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8(2), pp. 156–167, 2019. doi: 10.11591/ijai.v8.i2.pp156-167.

[29] Wickremasinghe B., Calheiros R.N., Buyya R.: CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications. In: *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pp. 446–452, IEEE, Perth, Australia, 2010. doi: 10.1109/AINA.2010.32.

[30] Zhao L., Saif M.B., Hawbani A., Min G., Peng S., Lin N.: A novel improved artificial bee colony and blockchain-based secure clustering routing scheme for FANET, *China Communications*, vol. 18(7), pp. 103–116, 2021. doi: 10.23919/JCC.2021.07.009.

Affiliations

Daisy Sharmah
University of Science and Technology Meghalaya, Department of Computer Science, India, e-mail: sharmah.daisy@gmail.com

Kanak Chandra Bora
University of Science and Technology Meghalaya, Department of Computer Science, India, e-mail: boopborabora@yahoo.co.in

Junumoni Khakhlari
University of Science and Technology Meghalaya, Department of Computer Science, India, e-mail: junumonikhakhlari608@gmail.com

Received: 28.09.2024
Revised: 23.01.2025
Accepted: 26.08.2025