International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2024/38 Volume 11, Issue 5, September – October (2024)

RESEARCH ARTICLE

An Efficient Load Balancing HBLBACO Approach Using Hybrid BAT and LBACO Algorithm in Cloud Environment

Shalu Rani
Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.
✉ bansal11shalu@gmail.com

Dharminder Kumar
Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.
dr_dk_kumar_02@yahoo.com

Sakshi Dhingra
Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.
sakshi24.dhingra@gmail.com

Received: 14 June 2024 / Revised: 25 August 2024 / Accepted: 12 September 2024 / Published: 30 October 2024

Abstract – Cloud computing has emerged as a crucial paradigm for delivering scalable and efficient services to many users. Load balancing in cloud environments presents several challenges, such as optimizing makespan, degree of imbalance, standard deviation, enhancing system performance and processing speed, and ensuring a reliable cloud infrastructure. These challenges are exacerbated by dynamic and unpredictable workloads, which can lead to uneven distribution of tasks and underutilization or overloading of resources. To address the challenges proposed by dynamic and unpredictable workloads, various load-balancing algorithms have been proposed. This work presents a novel approach called the HBLBACO (Hybrid BAT and LBACO) algorithm to balance the load on cloud, which combines the strengths of the Bat algorithm (BA) and the Load Balancing Ant Colony Optimization (LBACO) algorithm that is local optima and global optima respectively to achieve improved load distribution in cloud environments. To analyse the proposed algorithm, extensive experiments were conducted using CloudSim simulation environments. The experimental results demonstrate that the HBLBACO algorithm reduces makespan, degree of imbalance, standard deviation and maximized processing speed. It effectively adapts to dynamic workload changes and achieves a more balanced distribution of tasks across VMs, leading to improved system performance. The results confirm that the proposed approach outperforms 8%, 68%, 71%, 81% then LBACO, 2%, 21%, 43%, 96% then ACO and 53% ,96%, 98% then PSO algorithm in terms of makespan, degree of imbalance, standard deviation and processing speed respectively.

Index Terms – Cloud Computing, Cloud Load Balancing, HBLBACO Algorithm, BAT Algorithm, LBACO Algorithm, ACO Algorithm.

1. INTRODUCTION

Cloud computing technology delivers a wide array of services to clients worldwide through the utilization of highly scalable and virtualized resources over the internet. The primary goal is to offer global services with minimal costs and optimal performance. In order to enable a numerous amount of clients around the globe to share resources efficiently and receive high-quality service within a specific timeframe, it is crucial to handle client requests in an efficient manner that minimizes time and resource wastage. This underscores the significant need for effective load balancing approaches, which serve as a critical factor of any cloud services provider for their success [1][2]

Load balancing plays a pivotal role in ensuring that cloud nodes are evenly distributed, preventing situations where certain resources are overwhelmed while others remain underutilized. This imbalance can result in a decrease in the response time for tasks assigned [3][4].Load balancing serves as an efficient approach to distribute workloads across resources, thereby enhancing processing speed and makespan times. The objective is to maintain an stable distribution of

RESEARCH ARTICLE

work among cloud resources, preventing both overloading and under loading scenarios [5].

Traditional algorithms [6] are employed for addressing this issue; however, their effectiveness diminishes when confronted with intricate and large-scale problems. Metaheuristic algorithms, including ant colony optimization (ACO) [7] [8], particle swarm optimization (PSO) [9][10], artificial bee colony (ABC) [11][12], and genetic algorithm (GA) [13], have gained popularity for tackling NP complete problems. As the complexity of the problem increases, convergence process and the speed of metaheuristic approaches using random population tend to deteriorate, especially when dealing with a higher number of jobs. The utilization of an efficient scheduling approach that generates high-quality initial solutions for the initial population of metaheuristic algorithms leverages the power of these dynamic approaches, thereby mitigating their limitations in dealing with intricately randomized initialization problems [14]

In this work, a hybrid BAT and LBACO algorithm (HBLBACO) is proposed to distribute the VM loads in an efficient way. To enhance the effectiveness of workload balancing, HBLBACO combines the strengths of two existing algorithms, the BA and LBACO algorithm. HBLBACO decreases the overall makespan time, standard deviation, and degree of imbalance. Furthermore, it maximizes the processing speed of the algorithm. When compared to other algorithms, HBLBACO is shown to be more efficient in resolving the workload balancing problem in cloud environments.

1.1. Problem Definition

Traditional load balancing techniques often struggle with the complexity of cloud environments, where workloads can vary widely in terms of processing requirements, and resource availability can fluctuate. Additionally, these techniques may suffer from issues like slow convergence, premature convergence to suboptimal solutions, or an inability to adapt to dynamic changes in the environment.

To address these challenges, there is a need for an advanced optimization algorithm that can effectively explore the solution space, quickly comply to changing conditions, and find near-optimal load distribution across virtual machines (VMs). The goal is to develop an algorithm that can improve overall system performance by minimizing makespan, degree of imbalance, standard deviation, and maximizing processing speed of all available resources.

So, this research's objective is to provide a Hybrid Bat and LBACO algorithm that effectively addresses the problem of load balancing in cloud computing environments. The hybrid algorithm combines the exploration capabilities of the Bat Algorithm with the exploitation strengths of LBACO algorithm to achieve optimal load distribution.

The significant contribution of proposed work:

- This work proposes a novel hybrid HBLBACO algorithm that effectively combines the global search ability of the BA Algorithm and the local search refinement of the LBACO algorithm.
- The proposed approach is performed using CloudSim Simulation
- The hybrid algorithm is tested at different number of tasks by increasing them
- The proposed hybrid approach is compared with LBACO, PSO, ACO algorithms.

The remainder of this paper is organized as follows: Section 2 represents the literature review on workload balancing. Section 3 represents the workflow of proposed algorithm. Section 4 presents the performance evaluation of the proposed approach, including experimental results and discussions. Section 5 concludes the paper and summarizes the future scope of this work.

2. LITERATURE REVIEW

A considerable body of research has dedicated to address the problem of load balancing and obtaining optimal assignment solutions. These research efforts can be broadly classified into three categories of algorithms: traditional, metaheuristic, and hybrid algorithms [15]

2.1. Traditional Algorithms

These approaches function by leveraging established data regarding resources and tasks to compute their assessment criteria. Many of these algorithms depend on time of execution to efficiently delegate tasks to resources that reduces makespan and load deviation. An example of such an algorithm is Min-Min approach, which serves as basis for various balancing algorithms [16]. Min-Min approach [17] determines the completion time of tasks submitted to different virtual machines (VMs). Task having the shortest time of completion is then allocated to its respective VM. Afterward, the completion times of the remaining tasks on that VM are adjusted. This task is then deleted from the list of unassigned tasks, and the process repeats until all tasks are allocated. [18].

The Load Balance Improved Min-Min (LBIMM) approach [19] enhances upon the traditional Min-Min approach. Initially, Min-Min approach is run to generate the initial solution for the next phase. Subsequently, completion time of the smallest task from the most loaded resource is determined across all VMs. The makespan is determined by relocating the task to the VM with the min completion time for that

RESEARCH ARTICLE

particular task. This makespan is then compared to the one generated by Min-Min. If the makespan is shorter than that of Min-Min, the reassignment of task to resource is done where it originated, and the start time of resources is adjusted. This iterative process continues until no further reassignments can reduce the makespan. As a result, heavily loaded resources are relieved, while lighter load are utilized efficiently. While traditional approaches are straightforward to implement and can enhance makespan, certain may neglect load deviation, particularly in scenarios with significant differences in resource speeds. Additionally, they may not achieve optimal solutions, especially as the problem complexity or size increases.

2.2. Metaheuristic Algorithms

Meta-heuristic techniques can provide approximate optimal solutions to complex problems in reasonable time period. Scheduling tasks in a large search space is difficult, as there are many possible solutions and it can take a long-time span to give the optimal solution. In such cases, no single, well-defined method can solve the problem. However, finding a near-optimal solution is often sufficient. As a result, meta-heuristic techniques are often used to schedule in task scheduling. The different types of metaheuristic algorithms includes ACO, BAT , ABC and(LBACO [20].

Xin-She Yang developed standard bat algorithm, is motivated by the bats echolocation behavior. It introduces frequency tuning and is recognized as the primary optimization and artificial intelligence algorithm. Every bat in the algorithm is characterized by its velocity (v) and location (x) within a d-dimensional solution space or search. The location represents a solution vector for a specific problem. Throughout the iterative search process, the algorithm maintains a population of n bats, with the best solution found ($x*$) being stored and updated accordingly [21][22]

Mallikarjuna et al. presented a binary bat algorithm produced particularly to address the valve-point effect of economic load dispatch problem. Their study highlighted several advantages of their binary bat algorithm. Notably, the algorithm demonstrated rapid convergence during the initial stages and exhibited the ability to dynamically shift from exploration to exploitation as the optimality of the solution improved. These characteristics make the binary bat algorithm an advantageous approach for efficiently tackling the valve-point effect of economic load dispatch problem [23]. Jayswal et al. proposed the Fault Tolerance BAT algorithm as a solution to address challenges related to assurance, liability, and reliability in cloud computing [24], [25].

Zehua Zhang et al suggested the approach to maximize resource allocation and enhance system performance. The method provides a possible strategy for achieving efficient load balancing in cloud computing by combining ACO and network theory [26]. Fidanova et al. proposed a load balancing approach for grid computing using ACO and Monte Carlo techniques. The goal of the algorithm is to find optimal scheduling for a set of tasks on a grid computing system [27].

Kun Li proposed the LBACO approach for scheduling independent tasks in load balancing. The motive of LBACO is to reduce the makespan, or total time of execution of all tasks while also balancing the load evenly across all virtual machines (VMs). LBACO achieves this by using ACO approach inspired by the behavior of ants. In ant colony optimization, ants deposit pheromones on the ground as they travel. These pheromones act as a signal to other ants, indicating that a particular path is a good one to follow. LBACO uses a similar approach, but instead of pheromones, it uses a probability distribution to guide the ants [28].

2.3. Hybrid Algorithms

Hybrid load balancing algorithms involve the fusion of two scheduling algorithms to leverage the strengths of both [29]. This paper introduces some widely recognized hybrid algorithms and elucidates the rationale behind proposing a new algorithm. The HGA-ACO algorithm [30], for instance, amalgamates GA and ACO. In HGA-ACO algorithm, a randomly initialized GA is employed to generate the pheromone for ACO. Subsequently, ACO iterates to produce the optimal result. The best fit result from both approaches are combined through crossover to yield the overall best solution. Despite the approach focus on factors such as execution time, throughput, and response time, it does not specifically address the load balancing problem [31].

The OH_BAC (Osmotic Hybrid Artificial Bee and Ant Colony) approach, as introduced in [32], utilizes the osmosis technique to establish an energy-efficient environment. In this approach, the ABC and ACO collaboratively work to identify the optimal virtual machine (VM) for migration to best suited physical machine. Furthermore, the algorithm activates the best suited osmotic host between all physical machines to reduce power consumption.

Table 1 depicts the summary of the algorithms that we came across our literature review. It includes methodology used, algorithms used, advantages and disadvantages of the discussed algorithms.

The rationale behind selecting the hybrid approach, combining the Bat Algorithm with LBACO, stems from the need to solve the inherent challenges of load balancing in cloud computing more effectively than traditional methods. The Bat Algorithm excels in global search capabilities due to its exploitation of frequency tuning and velocity updates, which help in diversifying the search space and avoiding local optima. However, while the Bat Algorithm is powerful in exploration, it can sometimes suffer from slow convergence in fine-tuning solutions. ACO algorithm is known for its

RESEARCH ARTICLE

strong exploitation capabilities, where ants progressively refine their solutions based on pheromone trails, resulting in a more intensive search in areas of the solution space. This pheromone-guided search is particularly effective in fine-tuning solutions but can be limited in its ability to explore the broader search space, especially in dynamic and complex environments like cloud computing.

By hybridizing these two algorithms, the resulting method leverages the strengths of both: the global search capability of the Bat Algorithm and the local search refinement of ACO. This synergy allows the hybrid approach to not only explore a wider range of potential solutions but also to converge more quickly and accurately on optimal load balancing configurations. Compared to traditional load balancing methods, which may rely solely on heuristic or static rules, this hybrid approach offers a more dynamic, adaptive, and efficient solution. It is particularly advantageous in cloud environments where workloads are highly variable, and resource availability can change rapidly, making it imperative to have a robust and flexible optimization strategy that can adapt in real time.

Table 1 Summary of Literature Review Algorithms

Algorithm	Algorithms Used	Methodology Used	Advantages	Disadvantages
Min-MIn[17]	Greedy algorithm	Greedy algorithm selecting task with the min time of execution and assigns to the resource that can complete it the earliest.	Simple and easy to implement. - Provides a fast-decision-making process.	May lead to load imbalances. - Not optimal for dynamic environments.
LBIMM (Load Balancing Improved Min-Min)[18]	Min-Min algorithm with load balancing modifications	Extension of MIN-min with load balancing enhancements to better distribute tasks across resources.	Improved load distribution over MIN-min. - Better handling of heterogeneous tasks.	Increased complexity compared to MIN-min. - May still cause imbalance in highly dynamic environments.
ACO (Ant Colony Optimization)[2]	ACO algorithm	Nature-inspired algorithm mimicking the foraging behaviour of ants, using pheromone trails to find optimal paths (task assignments).	Good for dynamic and distributed environments. - Scalability to large-scale systems. - Finds near-optimal solutions.	Convergence time can be high. - Dependent on parameter settings for performance.
BAT[33]	BAT algorithm	Metaheuristic algorithm inspired by bats echolocation behavior, balancing between global and local search for solutions.	Efficient in finding global optima. - Can handle multi-objective optimization. - Good balance between exploration and exploitation.	Performance depends on parameter tuning. - May suffer from premature convergence.
ABC (Artificial Bee Colony)[34]	ABC algorithm	Nature inspired algorithm mimicking the foraging behaviour of honey bees, focusing on exploration and exploitation of the search space.	Effective in exploring the search space. - Suitable for complex optimization problems. - Good convergence rate	Can be slower in converging compared to other algorithms. - Performance is sensitive to parameter values.
LBACO (Load Balancing Ant Colony Optimization)[28]	ACO with load balancing strategies	- Hybrid algorithm combining ACO with specific load balancing techniques to enhance task distribution.	Combines advantages of ACO and load balancing strategies. - Adaptable to changing conditions in cloud environments.	More computationally intensive. - Requires careful parameter tuning.

RESEARCH ARTICLE

HGA-ACO (Hybrid Genetic Algorithm and ACO)[30]	GA combined ACO	Hybridization of GA for global search ACO for refined local search.	High efficiency in finding near-optimal solutions. - Exploits both global and local search capabilities. - Robust against premature convergence.	Complex to implement. - Higher computational overhead. - Requires balancing between GA and ACO components.
OH-BAC (Osmotic Hybrid Artificial Bee and Ant Colony)[32]	ABC combined with ACO	Hybrid approach combining the ABC for exploration with ACO for effective task distribution.	- Combines the exploration strength of ABC with the optimization efficiency of ACO. - Well-suited for dynamic cloud environments. - High optimization capability and load balancing efficiency.	Complex implementation. - Computationally expensive. - Requires careful tuning of multiple algorithm parameters.

3. PROPOSED HBLBACO ALGORITHM

The hybrid HBLBACO algorithm proposed in this paper combines the BAT and LBACO algorithms to achieve efficient cloud computing load. The proposed hybrid algorithm maximizes the benefits of both algorithms to boost load balancing effectiveness and performance in cloud computing. The LBACO algorithm is capable of local search, whereas the bat algorithm is capable of global search. The parallel processing capabilities of cloud computing environments can also be utilized by the hybrid algorithm to speed up the load balancing procedure. Algorithm 1 is the algorithm for hybrid HBLBACO Algorithm.

Begin

Step 1: Initialize BAT Population

Initialize the population of bats with random positions and velocities in the search space

Initialize the parameters for loudness, pulse rate, frequency, and maximum iterations

Step 2: Main Loop

While (current iteration ≤ max iterations) do

Step 3: Fitness Evaluation of Bats using LBACO

For each bat in the population do

Calculate the fitness of the current bat using the LBACO algorithm

End For

Step 4: Update Best Fitness Value and Best Solution

If (current bat's fitness is better than the best fitness found so far) then

Update the best fitness value

Update the best solution with the current bat's position

EndIf

Step 5: Update Loudness and Pulse Rate of Bats using Fitness Value

For each bat in the population do

Adjust the loudness and pulse rate based on the fitness of the current solution

End For

Step 6: Update Frequency and Velocity of Bats using Current and Best Solution

For each bat in the population do

Update the bat's frequency based on the current position and best solution

Update the bat's velocity using the updated frequency and the difference in the current position and the best solution

Update the bat's position based on its velocity

End For

End While

Step 7: Return best fit

Return the best result found

End

Algorithm 1 Hybrid HBLBACO Algorithm

The flowchart of the proposed HBLBACO algorithm is represented in Figure 1.

Step by step implementation of HBLBACO algorithm:

3.1. Initialization Phase

In this step, the algorithm is initialized with the cloud infrastructure and user requests. The cloud infrastructure consists of a set of virtual machines with different capacities,

RESEARCH ARTICLE

and the user requests consist of a set of tasks with different computational requirements. The hybrid HBLBACO approach is initialized to a specific number of iterations. Various random solutions are generated in the first iteration. The Bat Algorithm and the LBACO algorithm are also initialized with their respective parameters, such as the population size, pheromone levels, and heuristic information. The randomly generated solutions work as the input to the next phase. The initialization process is represented mathematically in Eq. (1)

$X = \{x_1, x_2, \ldots \ldots x_n\}$ … (1)

where each x_n represents a load balancing strategy.

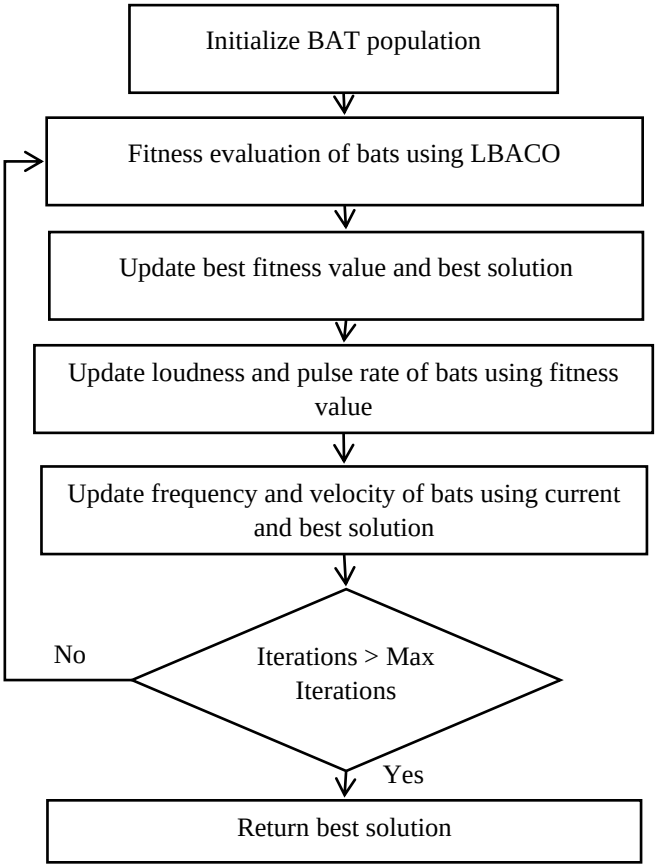

Figure 1 Flowchart of Proposed HBLBACO Algorithm

3.2. Fitness Evaluation

The fitness of the best generated solution is evaluated using LBACO algorithm. To evaluate the fitness of each bat solution using LBACO, each bat represents a potential solution, and the computation tasks are assigned to the resources based on the bat's solution. By evaluating the overall execution time and resource utilization across all tasks and resources, the performance of the bat's solution is evaluated. The fitness value increases as the overall execution time decreases and the tasks are spread more evenly among the resources. Equation 2 shows the mathematical representation of fitness evaluation.

$F(S)=\alpha(T(S)1)+\beta(U(S)$ (2)

Where: S represents the bat's solution, which is a potential assignment of computation tasks to resources,

T(S) is the total time of execution of the tasks based on the bat's solution,

U(S) is a measure of resource utilization efficiency, reflecting how evenly the tasks are distributed among the resources,

α and β are control parameters.

3.3. Best Solution Updating Phase

The best solution and best fit value are initialized to those of the first bat solution in this phase. The search procedure involves the initial best solution and best fitness value as a starting point. The fitness of each new solution is compared to the best fit value when the algorithm repeats the loop and creates new solutions for each bat. The best solution and best fitness value are modified to accommodate new solution if fitness of new solution is better than the current best fit value. By setting the initial best solution and best fitness value to the first bat solution's values, the algorithm ensures that the search process starts with a valid solution and that the best solution found so far is initialized correctly. This initialization step can also help to refine the efficiency of the search process by assigning a good starting point for the approach to survey the search space. This step is necessary to keep track of the best solution found so far and to update it when a better solution is found during the search process. Eq. 3 represents the mathematical formula this phase.

During the search process, for each new solution Si generated by the algorithm, the fitness F(Si)is calculated and compared with the current best fitness value Fbest. The best solution and fitness value are updated as follows:

If F(Si)>Fbest, then: Sbest=Si and Fbest=F(Si) (3)

Where: Si is the new solution generated during the search.

F(Si) is the fitness value of the new solution Si.

3.4. Update Loudness and Pulse Rate of Bats Using Fitness Value

This step updates each bat's loudness and pulse rate based on their current fitness. The loudness of a bat represents the intensity or amplitude of the bat's echolocation calls. In the context of the algorithm, it determines the step size of the bat's search process. A higher loudness value means that the bat can move further away from its current position in the search space. The loudness update is represented mathematically in Eq. (4)

RESEARCH ARTICLE

$B_{i(t+1)} = \alpha * B_{i(t)}$ ... (4)

Where $B_{i(t)}$ represents bat loudness, α is a control parameter.

The loudness of each bat is updated at each iteration of the algorithm based on its fitness value. If the evaluated fitness of the bat is better than the best evaluated fitness found so far, the loudness is decreased else increased.

The bats pulse rate determines the frequency of echolocation calls. In BAT algorithm it represents the search space of bats. Pulse update process is represented mathematically in Eq. (5).

if (rand > $r_{i(t)}$),

then $x_{i(t+1)} = x^k + \{\varepsilon * (L_B - U_B)\}$ … (5)

where $r_{i(t)}$ is the pulse rate of the bat, L_B and U_B are the lower and upper bounds of the search space, and ε is control parameter

If the pulse rate is high, then it will explore more search space. Based on fitness, pulse rate of bats is updated at each iteration.

3.5. Update Frequency and Velocity of Bats Using Current and Best Solution

This step updates each bat's frequency and velocity to explore the search space more effectively. This helps in finding the optimal solution. This process is mathematically represented in Eq. (6). The frequency is updated at each iteration.

$F_{i(t+1)} = F_{min} + \{(F_{max} - F_{min}) * r\}$ (6)

Where $F_{i(t)}$ represents bats frequency, F_{min} and F_{max} represents maximum and minimum frequencies, r represents random number between 0 and 1.

Velocity of bat represents the movement of bats in search space using its distance and direction. This process is mathematically represented in Eq. (7). It is updated by using the current and best solution. The updated velocity is added to the bat's current position to determine its new location.

$V_{i(t+1)} = V_{i(t)} + \{(x_{i(t)} - x^k) * A_{i(t)}\}$ … (7)

where $V_{i(t)}$ is the velocity of the bat, $A_{i(t)}$ is the loudness of the bat, $x_{i(t)}$ is the position of the bat, x^k is the global best solution

3.6. Termination and Output Phase

The algorithm will continue to run until the maximum number of iterations is reached. At each iteration, the algorithm will evaluate the fitness of each bat solution using the LBACO approach, update the loudness and pulse rate of each bat on the basis their fitness value, and update the frequency and velocity of each bat using the current and best solutions. After each iteration, the algorithm will check for the termination criteria. If it reaches to termination condition, the algorithm will terminate and return the best solution found so far. If the termination criteria have not been reached, the algorithm will continue to the next iteration and repeat the process.

4. RESULTS AND DISCUSSIONS

The HBLBACO algorithm is proposed using the Recloud for the purpose of evaluating the proposed algorithm. By implementing an effective layer of workload distribution and an appropriate environment for implementing various scheduling methods, the Recloud expands the current Cloudsim simulator. The outcomes of the proposed HBLBACO algorithm have compared with other work scheduling algorithms such as LBACO algorithm in order to analyze the performance of the algorithm. The efficiency of the HBLBACO algorithm was also evaluated in comparison to other comparable research which is covered in section 4.3.

4.1. Environmental Setup

Table 1 Simulation Parameters of Servers

Parameter	Value
Architecture	x64
Operating System	Linux
VMM	XEN
Time Zone	10.0
Memory Cost	0.05
Storage Cost	0.001

Table 2 Simulation Parameters of VMs

Parameter	Value
MIPS	9000
Processing Elements	1
RAM	512
BW	1000
VM Policy	TIME_Shared

Using simulation parameters for servers in Table 1, and virtual machines in Table 2, we performed extensive tests to compare the proposed algorithm's result in the load balancing problem with those of existing methods. Table 1 contains the simulation parameters of server in which VMM stands for Virtual Machine Monitor.

These parameters were used to determine the capabilities of the experiment's VMs and process. The proposed HBLBACO

RESEARCH ARTICLE

algorithm was developed with various number of tasks to analyze the four objectives:

1. Reduction in the makespan of the algorithm,
2. Minimizing the degree of imbalance,
3. Reducing the standard deviation value, and
4. Maximizing the processing speed of the algorithm

The HBLBACO evaluation tests were carried out using the parameters listed in Table 3.

Table 3 HBLBACO Algorithm Parameters

Parameter	Value
Population Size	100
Number of Iterations	1000
AMIN, AMAX	0, 5
RMIN, RMAX	0, 5
D	3
A	1.0
B	4.0
Γ	2.0
Rho	0.05

Table 4 Number of Tasks in Each Scenario

Scenarios	Number of tasks
Scenario one	100
Scenario two	300
Scenario three	600
Scenario four	1000

The algorithm starts with 100 random solutions, called population size. The number of iterations chosen was 1000. In the BAT phase, the minimum and maximum loudness values represented by AMIN, AMAX of bats were 0 and 5 respectively. The minimum and maximum pulse rate denoted by RMIN, RMAX of bats were 0 and 5 respectively.

The value of D was 3 which represents Cloudlet size. In LBACO phase, the values of α, β, γ and Rho were 1.0, 4.0, 2.0 and 0.05 respectively. Here α represents pheromone exponent, β represents computing capacity, γ represents load balancing factor and Rho represents pheromone trail constant. Based on the characteristics of the VMs, four experiments were carried out, as indicated in Table 4. The workflow's size will be modified in order to test the algorithm's capacity. Table 4 lists the number of tasks for each of the four situations.

4.2. Performance Analysis

Table 5 HBLBACO vs LBACO

Algorithm	Makespan	Standard deviation	Degree of imbalance	Processing speed
Scenario 1				
HBLBACO	35.626	0.9743	0.0757	1602
LBACO	37.3378	2.6747	0.1923	510
Scenario 2				
HBLBACO	105.6881	3.3101	0.1024	3870
LBACO	112.9734	7.3225	0.2243	764
Scenario 3				
HBLBACO	202.6696	2.0931	0.0285	7374
LBACO	222.6867	14.3082	0.2135	1454
Scenario 4				
HBLBACO	346.0248	6.5202	0.048	13922
LBACO	370.9586	20.2984	0.1797	2152

To evaluate the decrease in makespan and standard deviation, each of the four scenarios have been put into operation. When compared to the LBACO algorithm, the proposed HBLBACO approach minimizes degree of imbalance and maximizes processing speed. Table 5 lists the outcomes of the experiments that were conducted for each of the four situations.

The workflow's task number increased for each case. In Scenario 1, the search space is restricted, which speeds up the process of finding the best solution. On the other hand, scenario four shows an extensive number of tasks to increase the search area. The wide space makes it difficult for the optimization algorithm to locate the best solution. Table 5 outcomes show how the performance parameters of the HBLBACO and LBACO algorithms differ. However, there is improvement in HBLBACO algorithm compared with LBACO algorithm. The HBLBACO algorithm performs better when search space was expanded as in scenario four. This outcome is attributed to the fast convergence of solutions, which prevents unnecessary solution a variety.

4.2.1. Makespan

Upon evaluating the makespan improvement between the HBLBACO algorithm and the LBACO algorithm, it becomes evident that the former demonstrates a substantial enhancement of 8% compared to the latter. This improvement is clearly illustrated in Figure 2. The key factor contributing

RESEARCH ARTICLE

to this progress is the HBLBACO algorithm's consistent and strategic selection of the most suitable virtual machines (VMs) for executing tasks.

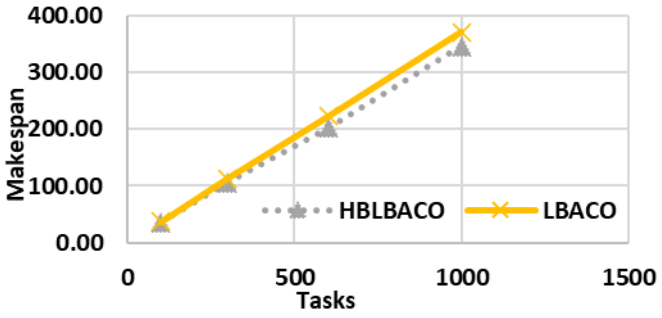

Figure 2 Makespan of LBACO vs. HBLBACO Algorithm

4.2.2. Standard Deviation

In terms of standard deviation, Figure 3 clearly demonstrates that the HBLBACO algorithm, which has been proposed, outperforms the LBACO algorithm by a significant margin of 71%. The reason behind this notable improvement is attributed to the efficient selection of virtual machines (VMs) by the HBLBACO algorithm.

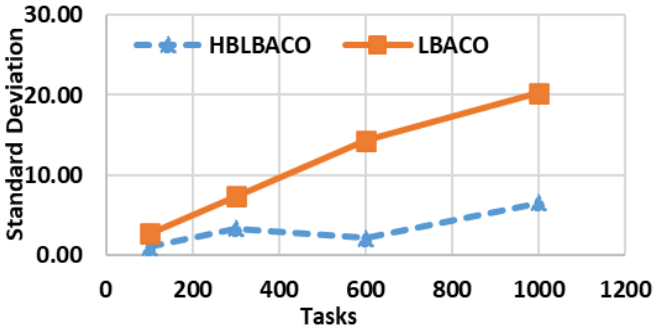

Figure 3 Standard Deviation of LBACO vs. HBLBACO Algorithm

4.2.3. Degree of Imbalance

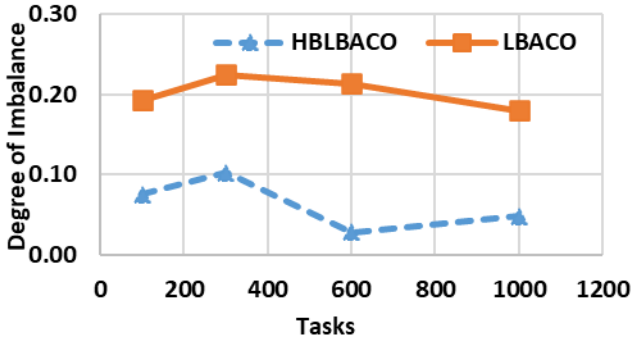

Figure 4 Degree of Imbalance of LBACO vs. HBLBACO Algorithm

With regards to the degree of imbalance, the findings presented in Figure 4 highlight a substantial 68% superiority of the proposed HBLBACO algorithm over the LBACO algorithm.

4.2.4. Processing Speed

The HBLBACO algorithm performs 81% better than the LBACO algorithm in terms of processing speed as shown in Figure 5. This significant performance is because the proposed algorithm uses suitable load distribution among VMs which leads to optimized speed of processing tasks.

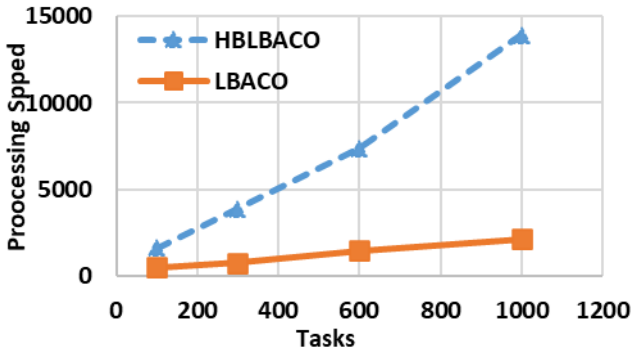

Figure 5 Processing Speed of LBACO vs. HBLBACO Algorithm

4.3. Comparison of Related Approaches

In order to compare different algorithms for scheduling workflow tasks, three specific algorithms were assessed: PSO algorithm and ACO algorithm. These algorithms were implemented based on their respective descriptions found in literature.

The outcomes indicate that the proposed HBLBACO algorithm not only enhances convergence to the optimal solution at a faster rate compared to the other algorithms, but also exhibits a higher quality of load balancing. These findings are elaborated upon in section 4.2 of the discussion.

Table 6 HBLBACO vs. PSO

Methods	Avg. Makespan	Avg. Standard deviation	Avg. Degree of imbalance
HBLBACO	170.5	2.05	0.057
PSO	364.5	108.12	1.45

The comparison between the PSO algorithm and the proposed HBLBACO algorithm was conducted with regards to three objectives: the makespan (total time to complete all tasks), the standard deviation (the measure of variability in task completion times), and the degree of imbalance (the distribution of tasks among resources). This comparison is presented in Table 6.

RESEARCH ARTICLE

Table 7 HBLBACO vs. ACO

Methods	Avg. Makespan	Avg. Standard deviation	Avg. Degree of imbalance	Avg. Processing Speed
HBLBACO	172.45	3.114	0.075	7094.5
ACO	175.58	5.456	0.0965	311.5

The comparison between the proposed HBLBACO algorithm and the ACO algorithm was conducted with respect to four objectives: the makespan (total time to complete all tasks), the standard deviation (the measure of variability in task completion times), the degree of imbalance (the distribution of tasks among resources) and processing speed (task completion in given timeframe). This comparison is presented in Table 7.

4.3.1. Makespan

Figure 6 shows that the HBLBACO algorithm outperforms 53% in terms of makespan than the PSO algorithm. Figure 7 shows that the proposed HBLBACO algorithm outperforms 2% better than ACO algorithm in terms of makespan.

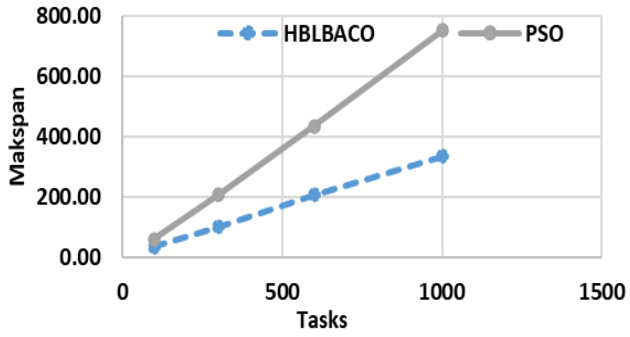

Figure 6 Makespan of HBLBACO vs. PSO

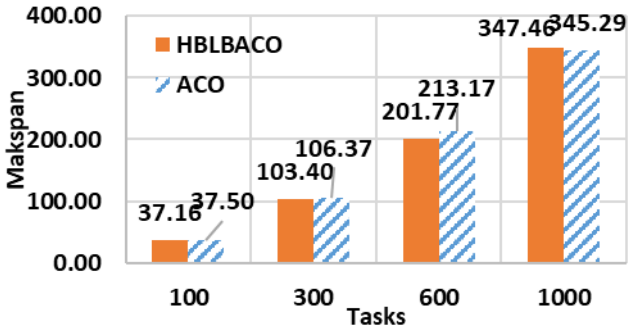

Figure 7 Makespan of HBLBACO vs. ACO

4.3.2. Standard Deviation

Figure 8 shows that the proposed HBLBACO algorithm outperforms 98% in terms of standard deviation than the PSO algorithm. Figure 9 shows that the proposed HBLBACO algorithm is 43% better than the ACO algorithm in terms of standard deviation.

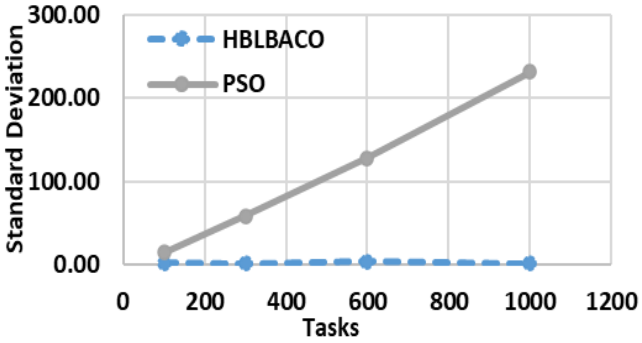

Figure 8 Standard Deviation of HBLBACO vs. PSO

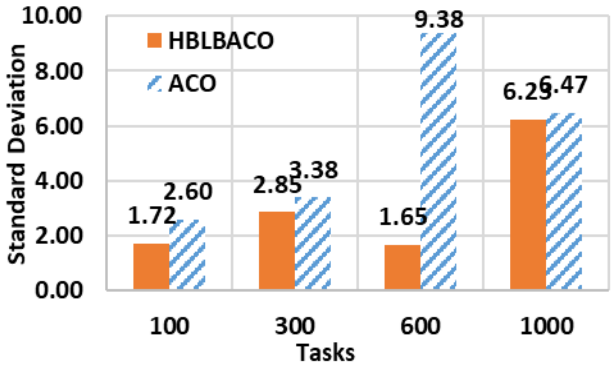

Figure 9 Standard Deviation of HBLBACO vs. ACO

4.3.3. Degree of Imbalance

Figure 10 shows that the proposed HBLBACO algorithm is 96% better than PSO algorithm in terms of degree of imbalance. Figure 11 shows the improvement of 21% in terms of degree of imbalance between the proposed HBLBACO algorithm and ACO algorithm.

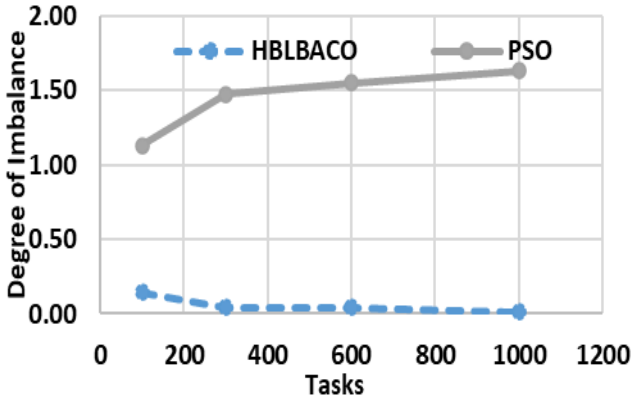

Figure 10 Degree of Imbalance of HBLBACO vs. PSO

RESEARCH ARTICLE

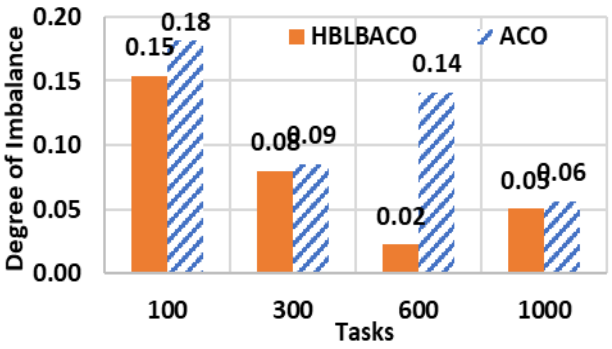

Figure 11 Degree of Imbalance of HBLBACO vs. ACO

4.3.4. Processing Speed

Figure 12 shows the improvement of 96% in terms of processing speed of proposed HBLBACO algorithm with ACO algorithm.

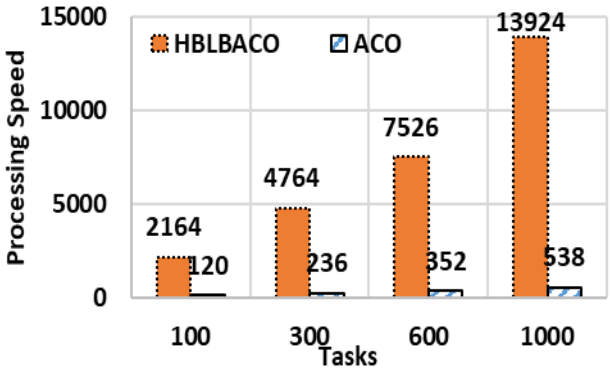

Figure 12 Processing Speed of HBLBACO vs. ACO

In essence, the proposed HBLBACO algorithm stands out due to its exceptional ability to combine suitable diversity and fast convergence. By effectively leveraging these features, it surpasses other algorithms by swiftly identifying the optimal solution, in any case of the number of workflow tasks involved.

5. CONCLUSION AND FUTURE SCOPE

The HBLBACO algorithm proposed in this paper is designed to address load balancing issues in cloud environments. The dispersion of load across available resources is referred as load balancing. To maximize resource use and prevent overloading particular capacities while maintaining others inactive, load balancing is used. The HBLBACO algorithm is implemented using the cloudsim simulator and compared with other known load balancing algorithms such as LBACO, PSO, and ACO. It is designed to distribute jobs over VMs with effective workload distribution to minimize makespan, standard deviation, and degree of imbalance, and increase processing speed. The outcomes of algorithm are compared with LBACO, PSO and ACO techniques. The HBLBACO algorithm optimizes makespan by 8%, 53%, and 2% compared to LBACO, PSO, and ACO, respectively. It also minimizes standard deviation by 71%, 98%, and 43%, and minimizes degree of imbalance by 68%, 96%, and 21%, respectively. The algorithm also enhances processing speed by 81% and 96% compared to LBACO, and ACO, respectively. The fitness function of the HBLBACO algorithm which balances cost and time has a big impact on the results. A mathematical formula called the fitness function is utilized in optimization problems to rate the effectiveness of solutions. To create a balance between these parameters, the algorithm utilizes the same weights for makespan, standard deviation, and degree of imbalance in the fitness function.

Future work can include extending the algorithm to heterogeneous environments with more than one data center, extending the workflow application distribution into two levels, using dynamic workflows to allow flexibility to users to change task attributes during runtime, and verifying the justification over real-time cloud environments. These extensions can enhance the scalability, flexibility, and efficiency of algorithm in various cloud environments.

REFERENCES

[1] Panwar, A. Singh, A. Dixit, and G. Parashar, “Cloud Computing and Load Balancing: A Review,” 2022 Int. Conf. Comput. Intell. Sustain. Eng. Solut. (CISES), 2022, pp. 334–343, 2022, doi: 10.1109/cises54857.2022.9844367.

[2] S. Rani, D. Kumar, and S. Dhingra, “A review on dynamic load balancing algorithms,” 3rd IEEE 2022 Int. Conf. Comput. Commun. Intell. Syst. ICCCIS 2022, pp. 515–520, 2022, doi: 10.1109/ICCCIS56430.2022.10037671.

[3] A. Hamidi, M. K. Goal, and R. Astya, “Load Balancing in Cloud Computing Using Meta-Heuristic Algorithm: A Review,” Proc. 2022 9th Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2022, pp. 639–643, 2022, doi: 10.23919/INDIACom54597.2022.9763131.

[4] S. Khare, U. Chourasia, and A. J. Deen, “Load Balancing in Cloud Computing,” in Proceedings of the International Conference on Cognitive and Intelligent Computing, 2022, pp. 601–608.

[5] S. T. Waghmode and B. M. Patil, “Adaptive Load Balancing in Cloud Computing Environment,” Int. J. Intell. Syst. Appl. Eng., vol. 11, pp. 209–217, 2023.

[6] M. A. Elmagzoub, D. Syed, A. Shaikh, N. Islam, A. Alghamdi, and S. Rizwan, “A survey of swarm intelligence based load balancing techniques in cloud computing environment,” Electron., vol. 10, no. 21, 2021, doi: 10.3390/electronics10212718.

[7] H. Xue, K. T. Kim, and H. Y. Youn, “Dynamic load balancing of software-defined networking based on genetic-ant colony optimization,” Sensors (Switzerland), vol. 19, no. 2, 2019, doi: 10.3390/s19020311.

[8] A. Gupta and H. S. Bhadauria, “Honey Bee Based Improvised BAT Algorithm for Cloud Task Scheduling,” Int. J. Comput. Networks Appl., vol. 10, no. 4, pp. 494-510journal, 2023, doi: 10.22247/ijcna/2023/223310.

[9] A. M. Manasrah and H. B. Ali, “Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing,” Wirel. Commun. Mob. Comput., vol. 2018, 2018, doi: 10.1155/2018/1934784.

[10] P. Jain and S. K. Sharma, “A Load Balancing Aware Task Scheduling using Hybrid Firefly Salp Swarm Algorithm in Cloud Computing,” Int. J. Comput. Networks Appl., vol. 10, no. 6, pp. 914–933, 2023, doi: 10.22247/ijcna/2023/223686.

RESEARCH ARTICLE

[11] H. Xing, F. Song, L. Yan, and W. Pan, "A modified artificial bee colony algorithm for load balancing in network-coding-based multicast," Soft Comput., vol. 23, no. 15, pp. 6287–6305, 2019, doi: 10.1007/s00500-018-3284-9.
[12] H. Saini, G. Singh, and M. Rohil, "Design of Hybrid Metaheuristic Optimization Algorithm for Trust-Aware Privacy Preservation in Cloud Computing," Int. J. Comput. Networks Appl., vol. 10, no. 6, pp. 934–946, 2023, doi: 10.22247/ijcna/2023/223690.
[13] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud Computing," Procedia Technol., vol. 10, pp. 340–347, 2013, doi: 10.1016/j.protcy.2013.12.369.
[14] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," J. Netw. Comput. Appl., vol. 128, pp. 64–77, 2019, doi: https://doi.org/10.1016/j.jnca.2018.12.010.
[15] W. Saber, W. Moussa, A. M. Ghuniem, and R. Rizk, "Hybrid load balance based on genetic algorithm in cloud environment," Int. J. Electr. Comput. Eng., vol. 11, no. 3, pp. 2477–2489, 2021, doi: 10.11591/ijece.v11i3.pp2477-2489.
[16] P. Neelima and A. R. M. Reddy, "An efficient load balancing system using adaptive dragonfly algorithm in cloud computing," Cluster Comput., vol. 23, no. 4, pp. 2891–2899, 2020, doi: 10.1007/s10586-020-03054-w.
[17] Y. Shi and K. Qian, "LBMM: A Load Balancing Based Task Scheduling Algorithm for Cloud," in Advances in Information and Communication, 2020, pp. 706–712.
[18] M. Haghi Kashani and E. Mahdipour, "Load Balancing Algorithms in Fog Computing: A Systematic Review," IEEE Trans. Serv. Comput., vol. 1374, no. c, pp. 1–18, 2022, doi: 10.1109/TSC.2022.3174475.
[19] S. Abdolhosseini and M. T. Kheirabadi, "Scheduling Independent Parallel Jobs in Cloud Computing : A Survey," vol. 11, no. 3, pp. 11–21, 2019.
[20] Z. Shafahi and A. Yari, "An efficient task scheduling in cloud computing based on ACO algorithm," 2021 12th Int. Conf. Inf. Knowl. Technol. IKT 2021, pp. 72–77, 2021, doi: 10.1109/IKT54664.2021.9685674.
[21] X. S. Yang, "A new metaheuristic Bat-inspired Algorithm," Stud. Comput. Intell., vol. 284, pp. 65–74, 2010, doi: 10.1007/978-3-642-12538-6_6.
[22] B. Xing and W.-J. Gao, Innovative Computational Intelligence : A Rough Guide to 134 Clever Algorithms. 2014.
[23] B. Mallikarjuna, K. H. Reddy, and O. Hemakesavulu, "Economic Load Dispatch Problem with Valve – Point Effect Using a Binary Bat Algorithm," ACEEE Int. J. Electr. Power Eng., vol. 4, no. 3, pp. 33–38, 2013.
[24] A. K. Jayswal, "Efficient task allocation for cloud using bat Algorithm," PDGC 2020 - 2020 6th Int. Conf. Parallel, Distrib. Grid Comput., pp. 186–190, 2020, doi: 10.1109/PDGC50313.2020.9315845.
[25] A. K. Jayswal and D. K. Lobiyal, "Fault Aware BAT Algorithm for Task Scheduling in Cloud," Proc. 2021 10th Int. Conf. Syst. Model. Adv. Res. Trends, SMART 2021, pp. 104–108, 2021, doi: 10.1109/SMART52563.2021.9676253.
[26] Z. Zhang and X. Zhang, "A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation," ICIMA 2010 - 2010 2nd Int. Conf. Ind. Mechatronics Autom., vol. 2, pp. 240–243, 2010, doi: 10.1109/ICINDMA.2010.5538385.
[27] S. Fidanova and M. Durchova, "Ant algorithm for grid scheduling problem," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 3743 LNCS, pp. 405–412, 2006, doi: 10.1007/11666806_46.
[28] K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization," Proc. - 2011 6th Annu. ChinaGrid Conf. ChinaGrid 2011, pp. 3–9, 2011, doi: 10.1109/ChinaGrid.2011.17.
[29] A. Hota, S. Mohapatra, and S. Mohanty, Survey of Different Load Balancing Approach-Based Algorithms in Cloud Computing: A Comprehensive Review, vol. 711. Springer Singapore, 2019. doi: 10.1007/978-981-10-8055-5_10.
[30] B. Wang and J. Li, "Load balancing task scheduling based on Multi-Population Genetic Algorithm in cloud computing," Chinese Control Conf. CCC, vol. 2016-Augus, pp. 5261–5266, 2016, doi: 10.1109/ChiCC.2016.7554174.
[31] K. Pradeep and D. Pravakar, "Exploration on Task Scheduling using Optimization Algorithm in Cloud computing," 2022 6th Int. Conf. Trends Electron. Informatics, ICOEI 2022 - Proc., no. Icoei, pp. 874–877, 2022, doi: 10.1109/ICOEI53556.2022.9777120.
[32] M. Gamal, R. Rizk, H. Mahdi, and B. E. Elnaghi, "Osmotic Bio-Inspired Load Balancing Algorithm in Cloud Computing," IEEE Access, vol. 7, pp. 42735–42744, 2019, doi: 10.1109/ACCESS.2019.2907615.
[33] A. Singh, A. Meyyazhagan, and S. Verma, "Nature-Inspired Computing: Bat Echolocation to BAT Algorithm," in Nature-Inspired Intelligent Computing Techniques in Bioinformatics, K. Raza, Ed. Singapore: Springer Nature Singapore, 2023, pp. 163–174. doi: 10.1007/978-981-19-6379-7_9.
[34] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm," Appl. Soft Comput. J., vol. 11, no. 1, pp. 652–657, 2011, doi: 10.1016/j.asoc.2009.12.025.

Authors

Shalu Rani received the B. Tech and M. Tech degrees in Computer Science Engineering from Guru Jambheshwar University of Science and Technology in 2019, and 2022, respectively. Her research interests are in the area of computer networking, including cloud computing, and Internet of Things. She can be contacted at email: bansal11shalu@gmail.com.

Sakshi Dhingra is an Assistant Professor in Department of Computer Science and Engineering at Guru Jambheshwar University of Science and Technology, Hissar, Haryana, India. She graduated from Punjab Technical University Jalandhar in 2010 and received a Master's Degree (Gold Medal list) in Computer Science and Engineering from Guru Jambheshwar University of Science and Technology in 2012. She has presented many papers in International Journals/Conferences. She has supervised many students at the M.Tech and MCA level. Her areas of interest are Image Processing, Soft Computing, and Remote Sensing. She can be contacted at email: sakshi24.dhingra@gmail.com.

Prof. Dharminder Kumar is currently working as a Prof. & Dean, Faculty of Science & Technology & NAAC Coordinator, Gurugram University, Gurugram-122003. He is a Retd. Professor from GJUS&T, Hisar-125001. He is the coordinator of the Technical Education Quality Improvement Program (TEQIP) of the World Bank. He also worked as Dean of the Faculty of Engineering and Technology, Head of the Department (HOD) of Computer Science and Engineering and founder Dean of Colleges. He has supervised many students at PhD and M.Tech. Level. His areas of interest include Data Mining and Computer and Communication Networks. He can be contacted at email: dr_dk_kumar_02@yahoo.com, dr.dk.kumar.02@gmail.com & dharminder.kumar@gurugramuniversity.ac.in.

RESEARCH ARTICLE

How to cite this article:

Shalu Rani, Dharminder Kumar, Sakshi Dhingra, "An Efficient Load Balancing HBLBACO Approach Using Hybrid BAT and LBACO Algorithm in Cloud Environment", International Journal of Computer Networks and Applications (IJCNA), 11(5), PP: 594-606, 2024, DOI: 10.22247/ijcna/2024/38.