International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/223310 Volume 10, Issue 4, July – August (2023)

RESEARCH ARTICLE

Honey Bee Based Improvised BAT Algorithm for Cloud Task Scheduling

Abhishek Gupta
Department of Computer Science and Engineering, Govind Ballabh Pant Institute of Engineering and Technology, Pauri, Uttarakhand, India.
abhi.g2010@gmail.com

H.S. Bhadauria
Department of Computer Science and Engineering, Govind Ballabh Pant Institute of Engineering and Technology, Pauri, Uttarakhand, India.
hsb76iitr@gmail.com

Received: 02 May 2023 / Revised: 15 July 2023 / Accepted: 06 August 2023 / Published: 31 August 2023

Abstract – Delivering shared data, software, and resources across a network to computers and other devices, the cloud computing paradigm aspires to offer computing as a service rather than a product. The management of the resource allocation process is essential given the technology's rapid development. For cloud computing, task scheduling techniques are crucial. Use scheduling algorithms to distribute virtual machines to user tasks and balance the workload on each machine's capacity and overall. This task's major goal is to offer a load-balancing algorithm that can be used by both cloud consumers and service providers. In this paper, we propose the 'Bat Load' algorithm, which utilizes the Bat algorithm for work scheduling and the Honey Bee algorithm for load balancing. This hybrid approach efficiently addresses the load balancing problem in cloud computing, optimizing resource allocation, make span, degree of imbalance, cost, execution time, and processing time. The effectiveness of the Bat Load algorithm is evaluated in comparison to other scheduling methods, including bee load balancer, ant colony optimization, particle swarm optimization, and ant colony and particle swarm optimization. Through comprehensive experiments and statistical analysis, the Bat Load algorithm demonstrates its superiority in terms of processing cost, total processing time, imbalance degree, and completion time. The results showcase its ability to achieve balanced load distribution and efficient resource allocation in the cloud computing environment, outperforming the existing scheduling methods, including ACO, PSO, and ACO and PSO with the honey bee load balancer. Our research contributes to addressing scheduling challenges and resource optimization in cloud computing, providing a robust solution for both cloud consumers and service providers.

Index Terms – BAT Load Algorithm, Bat Algorithm, Cloud Computing, Honey Bee Algorithm, Load Balancing, Resource Allocation, Task Scheduling.

1. INTRODUCTION

In recent times, cloud computing has been a critical revolution in network technology. It is an on-demand computing system that uses virtualization systems to deliver cloud resources to customers in the form of virtual machines over the Internet. The underlying concept behind distributing hardware and software resources is, cloud computing entails delivering associated services based on consumer demands [1]. The cloud computing architectural style consists of three different categories, the first of which is "Software as a Service" (SaaS). Moreover, the system can offer platform users software assets designed in the SaaS architectural style to make it easier for users to use. Platform as a Service (PaaS) is the second type, and it allows individuals to create their projects through the platform. Infrastructure as a Service is the final type (IaaS) [2]. Figure 1 depicts the cloud computing architecture.

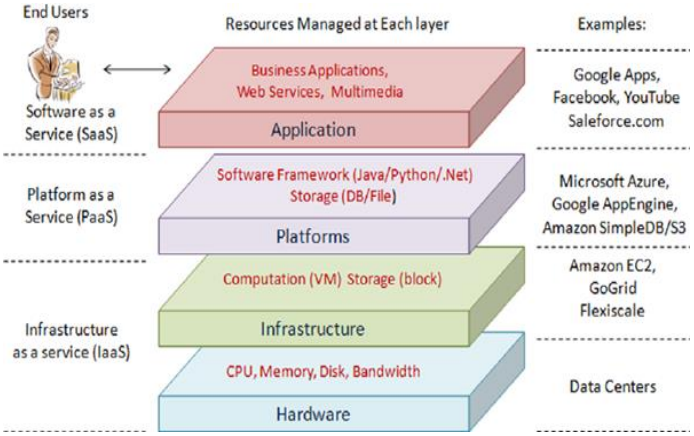

Figure 1 Architecture of Cloud Computing [3]

RESEARCH ARTICLE

Cloud computing, as a versatile innovation, is an excellent choice for organizations looking to build their private cloud. Users benefit greatly from cloud computing strategies because information can be accessed from any location at any time. This allows for quicker application retrieval with less downtime, allowing for efficient data recovery services. The cloud is easy to scale, allowing businesses to add or remove resources as needed. Cloud computing provides numerous advanced data security mechanisms, such as unlimited storage capacity and automatic Software Updates [4]. Cloud service providers offer a wide range of applications in the fields of art (Adobe Creative Cloud), business (Salesforce, Paypal), data storage systems and backup (Google G Suite), education (AWS, Tablets), entertainment (Vortex, PlayStation Now), management (Evernote), social networking sites (Yammer), and so on. Organizations can now utilize big data analytics to gather valuable information and cloud services for product design, testing, and deployment.

Cloud computing system offers numerous benefits to consumers, but several difficulties and issues persist. Because time is short in these private clouds, the final goal is to maximize resource utilization while also providing an assured service to clients. The problem of job scheduling is considered one of the most recognized topics in a cloud computing environment. Job scheduling entails assigning the user's tasks to the appropriate resources made available by the service provider [5]. It is the formula of assigning machines VMR= {VMR1, VMR2 …, VMRm} to activities CA= {CA1, CA2 …, CAn} over time for accomplishing the chosen goal. In a cloud computing environment, job scheduling is necessary to achieve load balancing. The primary objective of this study is to introduce a new load-balancing approach in a cloud system. Load balancing is essential for allocating load to cloud servers for getting higher throughput and better resource utilization. The goal of it is to keep all servers occupied and to make greater use of resources effectively [6]. An efficient load balancing algorithm takes into account variables including the type of service, load requirements, and processing capacity. The various load balancing and scheduling techniques used in cloud computing are depicted in Figure 2.

Load balancing methods are distributed into two types: dynamic and static scheduling techniques. The dynamic scheduling technique is ideal for real-time task assignments in which the system will allocate tasks without having to know the job completion time [7]. Using a priori known information, static scheduling approaches like min-min, max-min, and Suffrage algorithms schedule activities more efficiently than dynamic scheduling strategies [8]. As a result, static scheduling methods outperform dynamic ones in terms of performance and load balancing. However, a few load balancing schemes wherein activities accomplished by deprived nodes reduce a significant number of heterogeneous tasks exist.

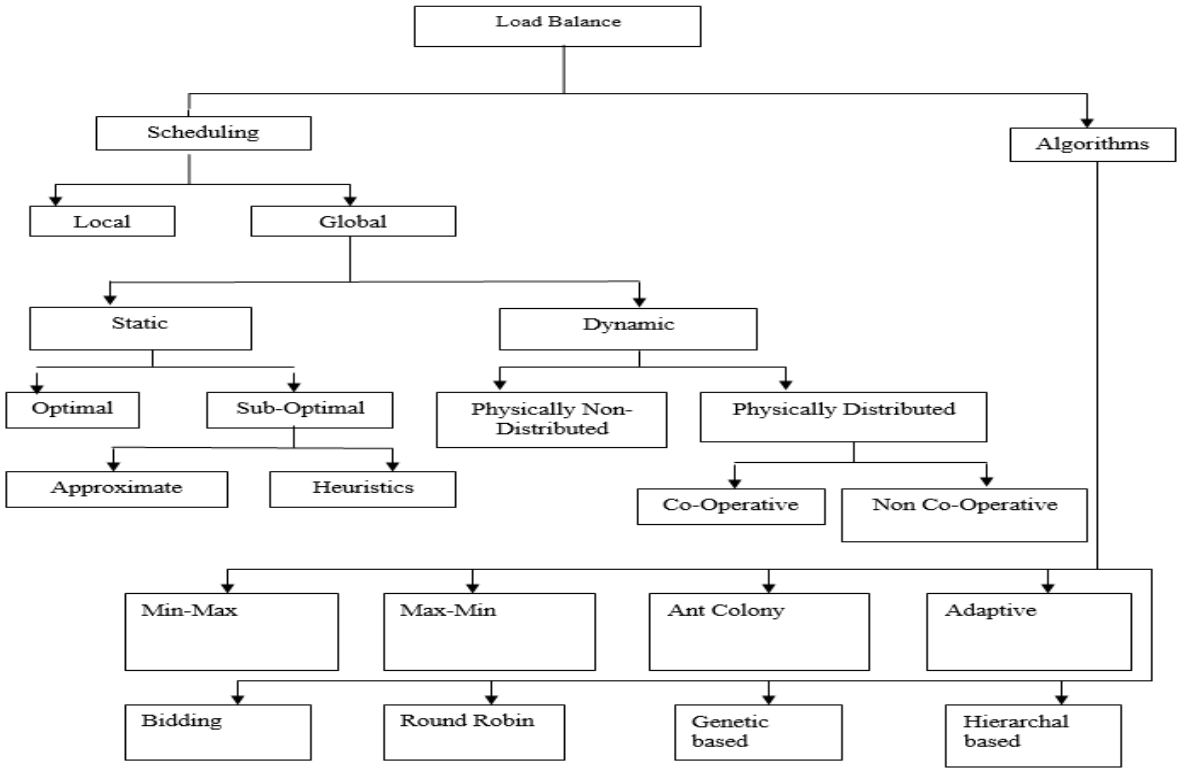

Figure 2 Different Load Balancing Scheduling Algorithms [9]

RESEARCH ARTICLE

The motivation of the research is the development of a job scheduling procedure that takes into account and as a result, a honey bee algorithm is presented to assign tasks more effectively in various cloud computing environments. The performance of task scheduling algorithms such as ACO, PSO, and BA are compared with each other. The proposed task scheduling algorithm outperforms previous algorithms in terms of time, load balancing, and resource utilization. It has the potential to boost the scheduling of cloud computing.

The following is how this study is set up. Related work is included in Section 2. Section 3 describes conventional methods such as ant colony optimization, particle swarm optimization, and the BAT algorithm. The proposed tasks are fully described in Section 4. Results in numerical form for the suggested task are shown in Section 5. In Chapter 6, which wraps up the investigation, findings are presented along with suggestions for potential future extensions.

2. LITERATURE REVIEW

In this part of the paper, the research progress on accomplishing high computation in cloud applications based on various business needs, job scheduling, resource utilization, load balancing, and so on is presented.

R. Kumari et al. [10] evaluate the effectiveness of the makespan and CPU consumption measures; we offer a job scheduling approach for a variety of independent jobs in our study by combining the Max-min algorithm and particle swarm optimization. The conventional PSO technique takes some time to locate the overall optimal solution because the initial population was generated randomly. This means that the max-min algorithm produces outputs of potential task sequences that can be assigned to virtual machines depending on appropriateness criteria. The PSO algorithm uses the findings as input. CLOUDSIM is used in the suggested method to locate the best answer. The proposed model's completion time parameter is 5.01% better and its CPU utilization rate is 3.63% higher than that of conventional methods.

By combining the gravitational search concept with ant colony optimization, S. Rani et al. [11] devised a hybrid technique. This not just managed to avoid the early convergence problem of ACO, but also achieved the distributed nature that GSA did not have. The outcomes are obtained using the CloudSim toolkit. It is demonstrated that the proposed algorithm reduces task completion time and allocates equivalent loads to every virtual machine, resulting in load balance and reduced resource utilization. It can also assess machine capabilities and efficiently assign tasks to them, boosting their efficiency. The management interface for storage, memory, virtual machines, and bandwidth is one of the features included in the CloudSim architecture layers that facilitate simulation and modeling of the cloud environment. It also provides hosts, dynamic system state, and management of application execution for the VM. In a grid computing environment, it is more difficult to distribute resources to several workloads than in network computing systems.

A novel hybrid CR-AC technique was demonstrated by A.A. Nasr et al. [12] to fix a scheduling problem with a deadline constraint. A modified chemical reaction optimization technique discovers an optimal outcome in minimal time complexity under deadline constraints, and the ACO algorithm minimizes overall cost and enhances the quality of the solution. The CR-AC arranges a significant amount of tasks into VMs while taking into account multiple resource constraints, improving the overall system's performance. The CR-AC method generates high-quality solutions at a lower cost than CRO, ACO, modified PSO, and cost-effective genetic algorithm (CEGA), making it more efficient than those methods. This particular ACO algorithm decreased overhead issues, but the end result was a subpar reaction time. These algorithms, however, are typically rigid and unable to adapt to dynamic changes in the properties while being executed.

The ant colony system is reconsidered by Xiang et al. [13], and a novel approach for ACS-based job scheduling is presented. To accomplish scheduling, greedy minimum planning for machine allotment is implemented at the same time. Investigations on synthetic and real-world application graphs show that Greedy-Ant surpasses the current algorithms speed by 18%. Xiang, B. et al. [14] in their study suggested an upgraded version of the bee colony algorithm with the goal of security and quality optimization of arranging tasks in a cloud system. The job is allocated to the most suitable virtual machine based on the customer's adequate security level and service quality policies. Each data center's storage of hive tables helps to balance the demands on virtual machines while lowering costs, makespan, and security threats. This algorithm outperforms the traditional ABC algorithm in terms of security and cloud user service quality. Based on this study's explicit explanation of the ABC algorithm's customizable fitness function, a workflow program may be selected that has a minimum cost, a minimum makespan (completion time), or any value in between. To solve bottleneck issues and achieve a shorter makespan, a heuristic was created.

In Kruekaew et al. [15] research, a methodology that utilizes heuristic job scheduling with Artificial Bee Colony for VMs is designed to enhance data load balancing in cloud networks. The proposed approach is compared to ACO, PSO, and improved PSO techniques. The research was carried out using four different datasets to assess HABC effectiveness. HABC with Largest Job First heuristic algorithm (HABC LJF) provides the best scheduling efficiency. It introduced a more advanced PSO based approach by defining the cost vector and limit of the Initialization Solution and the Find Solution Space

RESEARCH ARTICLE

in the Exist Solution Space. Although this strategy has proven successful, it is rather complicated.

Authors examine the load balancing problem in multicast using network coding in the study of Xing et al. [16]. The food source initialization (FSI), NSL-based selection (NSL-S), and neighborhood search (NS) schemes are all extensions of the modified artificial bee colony algorithm (MABC). The three extensions have demonstrated their ability to significantly boost the overall efficiency of the proposed technique. In terms of solution quality, MABC outshines binary ABC (binABC), binary-coded ABC (BABC), and modified discrete ABC (MDisABC) algorithms. This algorithm improves ABC optimization, reduces average operation time, increases resource utilization ratio, and competently supplies appropriate resources to user job. However, it is ineffective for large-scale optimization.

Rani et al. [17] describe the Modified-HBB-LB approach, which employs modified-Blowfish optimization to retrieve cloud resources employed at work to balance burden. In comparison to HDLB, the migration effort is reduced by 30%, 25%, and 20%, respectively, by bee load balancing and Modified-HBB-LB. In terms of completion time, completion time, and reaction time, Modified-HBB-LB maintains a high degree of efficiency of 3–5%.

Jeyalaksshmi et al. [18] proposed a Modified Round Robin and Modified Honey Bee Methods for efficient load balancing depending on foraging interaction to regulate load. Jobs taken from overcrowded VMs are referred to as honey bees. MHBA removes jobs from virtual overloaded devices and routes them to previously used VMs. MRRA is employed for jobs that have no set priority. It selects available VMs for higher-priority tasks based on each VM's current job. The suggested methodology reduces reaction time and data center processing time. This algorithm's key discovery was that jobs were assigned to virtual machines in order to reduce waiting time and increase system throughput.

Babu L.D. Dhinesh et al. [19] created a new program that employs the LBMM (Load Balance Min-min) approach to distribute jobs among resources and chooses the best resource based on bee foraging behavior to finish the task in the lowest amount of time. The ABC method, which was modeled after how bees behave, helps balance out VM loads (overloaded & underloaded) and increase throughput. The suggested method can be utilized for thorough resource planning since it maximizes the use of available resources, operates at its peak efficiency, responds quickly, is scalable, and is durable.

D. Chaudhary et al. [20] described job scheduling for data processing in cloud computing using cloudlets and VMs. PSO approach for load balancing is focused on particle fitness values and forces acting on them. The proposed New PSO approach implemented a novel cost estimation method on cloudlets for overall cost optimization and produces more accurate and cost-effective results than PSO. Future work will include developing a new objective method for more cost optimization in the cloud by utilizing various emulators and hosts.

The authors advocate secure scheduling protocols and smart, variable neighbourhood PSO algorithms for cloud computing to make the best use of resources. With little task scheduling decision time, J.A.J. Sujana et al. [21] offers automatic IaaS provisioning. It offers a compromise between security, the shortest possible job time, and cloud task cost. The encoding methods SPSO and SVNPSO use were created for particle encoding to tackle multi-objective problems. Because they require fewer repetitions than earlier techniques, these algorithms perform better. To assess the effectiveness of the metaheuristics, a workflow scheduling algorithm that considers security and cost was chosen.

Authors introduce Cost-effective fault-tolerant scheduling in P. Guo et al. [22] to decrease execution costs while attempting to ensure that more jobs meet deadlines. The backup approach is used to obtain fault tolerance. CEFT iteratively optimizes job mapping and resource management with PSO. To boost the deadline guarantee ratio, the rescheduling technique is used. To make the scheduling process more flexible and less dependent on local optimal, CEFT makes use of PSO's benefits.

In their hybrid bat algorithm proposal, Zheng et al. [23] approach avoids becoming stale in local minima by classifying bat populations. The search volume and pulse firing rate can be increased by using back propagation systems that leverage paired gradient mechanisms and mean square error. Furthermore, the Levy Flight-based random walk strengthens the algorithm's capability for global navigation while improving the best outcome. ACO, GA, PSO, and CSA algorithms are outperformed by the bat algorithm in terms of throughput, imbalance, and completion time.

Raj et al. [24] proposed a Modified Bat Algorithm that makes it possible for load balancing among VMs. Both "Overload Optimal VM" and "Balanced VM" are MBA variations. In comparison to a traditional BA, the MBA offers services that are both efficient and have unmistakable advantages. The work laid down by Barzegar et al. [25] provides PSO Bat-Greedy, a hybrid technique that decreases cost and time while increasing resource consumption. Resource utilization enhances by 15% and 5%, respectively, when compared to PSO and PSO-Bat algorithms.

Gu et al. [26] presented an Optimization heuristic algorithm based on the bat algorithm to schedule workflow in cloud network environments. By contrasting all local optimal solutions, EATTO determined the best global solution. It

RESEARCH ARTICLE

reduces the energy usage and processing time of complex scheduling jobs by increasing output without compromising the Quality of Service.

To choose the best collections of servers with quick convergence functionality, the suggested LB-RC method by Adhikari et al. [27] used clustering with Bat optimization. This lessens job execution time and duration while achieving the deadline. LB-RC balances the load of the cloud data center over time and efficiently utilizes the resources. Studies on two distinct synthetic datasets reveal that the novel approach performs better than conventional algorithms in producing work deployment plans and is more cost-effective to execute while meeting QoS restrictions. It primarily emphasizes system throughput. Using interactions between co-allocated jobs, it chooses the host with the highest throughput.

The Bat algorithm detects prey using echolocation to create a load-balancing framework. The suggested approach is based on the Naive-Bayes classifier, which is used to categorize VMs, which are then used to upgrade migrated tasks with data used by the bats. Based on the priority list in the queue, jobs from heavy-loaded VMs migrate to light-loaded VMs. The job is considered independent and non-preemptive by Load Balancing BA, and Quality of Service is only kept as a priority. It outperforms conventional methods like Dynamic load balance and round robin in terms of response, completion, and migration times presented by Ibrahim et al [28].

The well-known NP-complete problem of cloud computing load balancing for multifunctional scenarios is one issue with mobile cloud computing advances. Maximizing throughput, minimizing span, and conserving energy are among the most crucial objectives. To evenly distribute the system's load, a load balancer is necessary. To support priority-based scheduling, load balancers use meta-heuristics. We employed the "Simple BAT Algorithm Based on Bees" in this investigation because it is excellent at balancing load while utilizing available resources. To increase the capacity of virtual computers in data centers, the metaheuristic algorithm utilized in this study is an improvised BAT algorithm based on honeybees. The bee-based improvised BAT algorithm is adaptable, straightforward, and simple to use. Additionally, a major problem is successfully addressed. The same holds for the best ways to deal with complicated issues and offer reaction times.

3. TRADITIONAL ALGORITHMS

3.1. Ant Colony Optimization

Using this approach, we need to converse to every possible arrangement while moving across parameter space in search of the best configurations. Ants lay out pheromones to guide one another toward resources while assessing their fitness. To help additional ants discover better solutions throughout subsequent reproduction cycles, the reenacted "ants" similarly record their places and the nature of their responses. As seen in Figure 3, a fixed set of ants are first produced.

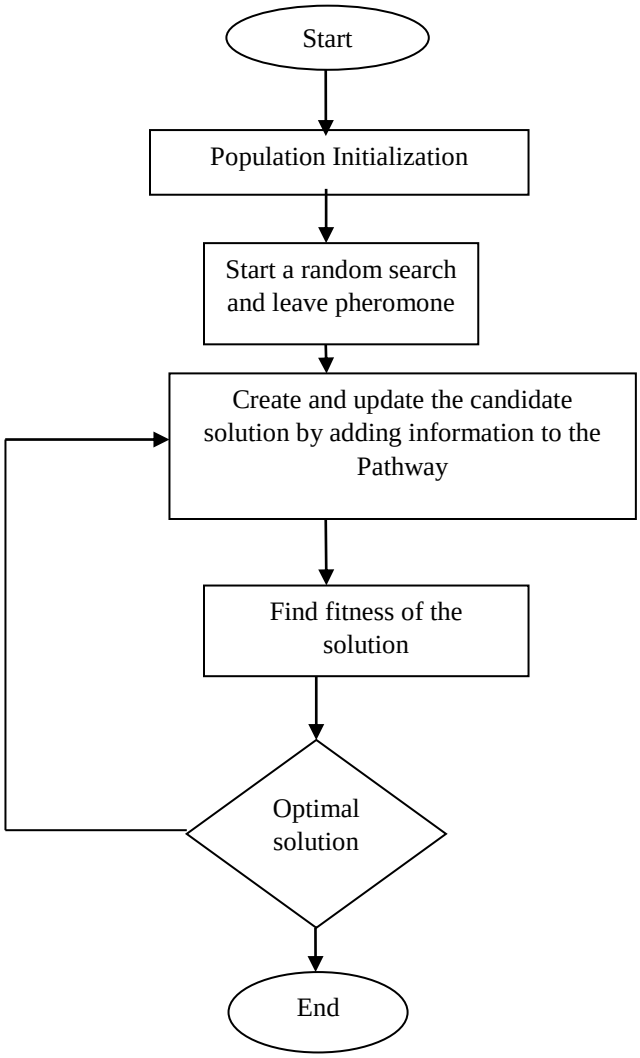

Figure 3 Ant Colony Optimization

Each secretes pheromones by the caliber of the meal. Following that, in an ongoing effort to discover a fix, it gathers and iteratively updates information on all ants with altered pheromones. The pheromone trail connects all edges and shortens to become the shortest path if pheromone deposition is vigorous. During their quest, other ants can use the pheromone trail as a repository. The last update of the solution space occurs for all ant pheromone trajectories [29]. Algorithm 1 depicts the Ant Colony Optimization.

1. Initialize $\tau\psi$ ant movement and $\eta\psi$, $\forall(\psi)$ pheromones.
2. Repeat step 2 for each ant in state k.

RESEARCH ARTICLE

3. Determine the state to move into with a high degree of probability.
4. Till ant k has finished solving its problem.
5. end for loop
6. Do for each ant movement (τιψ)
7. Calculate the fitness function
8. The trail matrix should be updated.
9. End for
10. If not, proceed to step 2 (end test).

Algorithm 1 Ant Colony Optimization ACO

3.2. Particle Swam Optimization (PSO)

Particle swarm optimization constantly tries to enhance scheduling arrangements to a certain percentage of value to address difficulties. By creating a population of potential assemblies, here called particles, and moving these particles to positions in the search space according to a simple formula and motion of the molecule, it solves a problem. A population collection of particles is initially formed with information about their location and speed. The particle's initial velocity is chosen at random, and it is adjusted later in the battle based on each participant's intelligence. This pbest information is replaced with the global best information, which is the fighting's leading particle. Moreover, it just needs a small number of parameters and has a rapid convergence rate [30]. As seen in Figure 4, a fixed set of PSO is first produced.

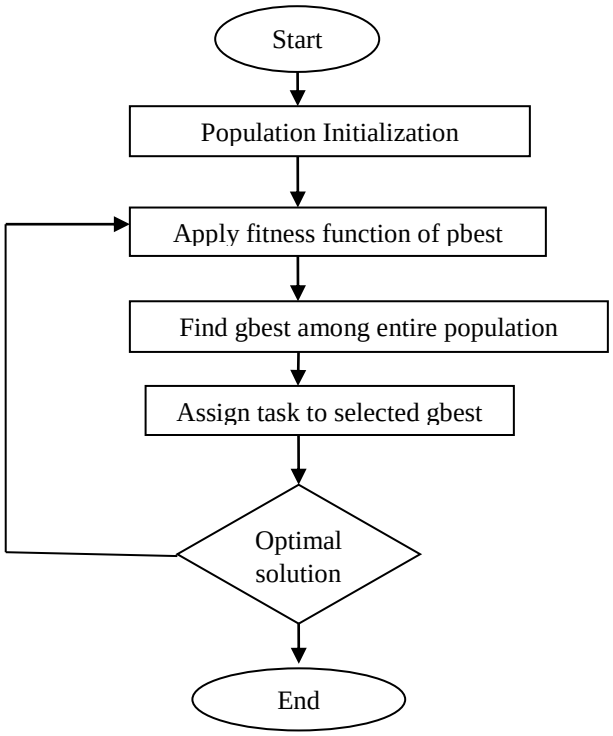

Figure 4 Particle Swam Optimization

Algorithm 2 depicts the Particle Swarm Optimization.

1. Set the particle's dimensions to d
2. Set the particle population's initial position and velocities at random.
3. Determines each particle's fitness value.
4. Compare the fitness value of the particle with the pbest of the particle. Set pbest to the current value and location if the current value is higher than pbest.
5. Compare the particle's fitness value to the global gbest. Set gbest to the current value and location if the current value is above it.
6. Update the particle's position as well as its velocity
7. Repeat from step3until stopping criterion is met

Algorithm 2 Particle Swam Optimization (PSO)

3.3. BAT Algorithm

An optimization approach having a random probability distribution that can be statistically analyzed is the BAT behavior-inspired optimization algorithm. It is based on how BATs approach their prey, which is similar to spotting machines with less load. Uncertain needs for qualitative notions and their numerical representation are a reflection of fuzziness, unpredictability, and their interrelationship. By applying the BAT method, the waiting time for a job is reduced as the convergence rate rises. The simulation demonstrates that the BAT method successfully optimizes functions. To increase the pace of convergence and accuracy, characteristics including population information, communication mechanisms, and random flying of BATs are explored and exploited. BAT prey is successfully located via echolocation. BAT uses echolocation to determine the location, range, and direction of its prey [28]. As seen in Figure 5, a fixed set of BAT is first produced. Algorithm 3 depicts BAT Optimization.

1. Initialize the P_i and V_i bat populations.
2. Explain the pulse frequency PF_i at P_i.
3. Establish the loudness L_i and pulse ratesPR_i.
4. for (iterations<Max_iterations)
5. Produce the solutions by modifying the frequency, velocities, and positions.
6. Use the fitness function to evaluate the solutions.
7. if randd >rr .
8. Select solution from the best solutions.
9. Produce a local solution based on the solution chosen from the best.

RESEARCH ARTICLE

10. If
11. Create a novel solution by randomly flying
12. If (randd Ai & f(Pi) & f(P*))
13. Agree the new solutions
14. Boost PR_i. lower F.
15. end if
16. Determine the top P* end currently available by ranking the bats.
17. end while
18. Return Bat_Solutions

Algorithm 3 BT Algorithm

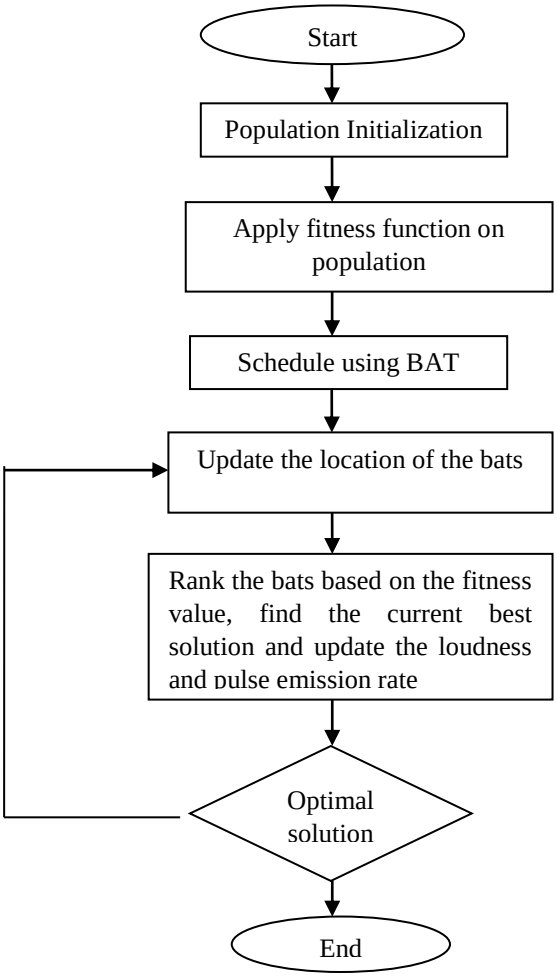

Figure 5 BAT Algorithm

4. PROPOSED WORK

In cloud computing environments, load balancing between virtual machines (VMs) is a challenge that this white paper aims to address. The proposed Honey Bee based improvised BAT algorithm, however, is prone to premature convergence when trying to find a workable solution to a particular problem since it is so readily trapped in local optima. An algorithm inspired by the Bat Algorithm was devised and implemented as a result of the suggested method. To determine the optimum solution, it uses magic to compute the distance between the artificial bat and the answer. The classic bat algorithm has been enhanced with an emphasis on parameter initialization, parameter updating, hybridizing with other methods, and transferring continuous space problems to binary space problems. In this research study, we suggest a method for calculating distance, which is similar to the approach taken by actual bats while shooting at their food. We also applied a modified version of the BAT method to load balance on the cloud. The four essential steps of the modified bat algorithm are parameter initialization, calculating fitness values, choosing the best virtual machine, and preventing VM overload. Here, the job is represented by the bat, and the target or bait is represented by the virtual machine. Bats hunt for items that are high in energy. More fitness values are correlated with higher energy levels, and more fitness values are correlated with lower distance values. Jobs are consequently dispatched to virtual machines that are located closer to the customer. A load balancing algorithm based on bees is proposed using the BAT load algorithm. The basic goal is to evenly spread the workload across numerous network links so as to prevent resource under- and overuse. By allocating incoming activities to virtual machines (VMs) that satisfy two requirements, this is accomplished. The difference between the virtual machine's processing time and the mean processing time of all virtual machines is less than a threshold, and the number of jobs it is currently processing is less than the number of jobs being processed by other virtual machines. Traditional load balancing techniques suffer from a number of drawbacks in cloud systems because of the shifting dynamics of the workload, host overload, and throughput reduction. In order to increase resource utilization and execution speed, this paper proposes a load balancing method that can seamlessly distribute dynamic workloads among all cloud hosts. Distribute all accessible virtual machines with incoming operations. The algorithm distributes tasks to the VM with the least load in order to ensure fairness and prevent congestion, using the difference between this VM's processing time and the average processing time of all VMs as the threshold. To avoid VM overutilization and hunger during task allocation, the proposed algorithm's key limiting factor for VM processing time variation. The standard deviation for preserving load balance throughout the system also has a significant impact. In cloud computing, the main issue is how to allocate resources for tasks efficiently when choosing the best virtual machine. Before choosing the right virtual machine, it is important to consider the resources required to perform each task and the available resources, or available

RESEARCH ARTICLE

virtual machines. In this paper, a comparison of three optimization algorithms is conceptualized, and boosted the performance of those algorithms with honey bee load balancing to provide effective cloud scheduling and load balancing. The first approach enhances the effectiveness of cloud scheduling and load balancing by combining the ant colony optimization algorithm and the bee algorithm. The algorithm schedules jobs using the ant colony optimization technique, and it calculates the load using the bee algorithm. The goal is to reduce a given job schedule's makespan and processing costs while enhancing load balancing based on workload. The proposed hybrid solution combines the Honey Bee load balancer with ACO algorithm schedules to create a new, efficient hybrid strategy. The second algorithm combines the particle swam optimization algorithm with honey bee load balancing. PSO offers a population-based search method that moves particles around in the search space to find the optimal solutions. After each iteration, each particle adjusts its position based on the best point encountered by itself and experience from the nearby particles. Each population is a schedule of virtual machines and tasks. After reaching maximum iterations best load-balancing optimized solution is generated. The third algorithm associates the BAT algorithm with the Honey Bee load balancer. The behavior of the bat algorithm defines adjustments in its characteristics, including position, velocity, frequency, loudness, and pulse emission rate. In this method, bats fly around a cloud in pursuit of prey as their velocity, frequency, and wavelength are varied. To find prey and avoid obstacles, it uses the echolocation of sonar pulses. The prey in this symbolizes tasks. Each bat in this finds the task that is closest using the fitness function for each object, finds new fitness using the optimization feature, and assigns the assignment. In order to determine which resource within the cloud network is best ideal to use, the Honey Bee load balancer is used. To distribute the load in the cloud system in an effective and scalable way, load balancing is a critical concern. Moreover, it guarantees that each computing resource is allocated properly and equally. Response time is a crucial challenge for every engineer to create an application that can maximize total throughput in the cloud environment, whereas all the existing algorithms that have been studied primarily consider the reduction of overhead, reducing migration time, enhancing performance, etc. Many traditional methods which lack optimal scheduling and load balancing lead to more processing costs. The proposed method, however, is more efficient in managing the cloud resource needs and reduces the response time for specific workloads. Figure 6 depicts the proposed flowchart. Algorithm 4 depicts the Honey Bee Load Balancer algorithm.

1. VmList=Optimized_Solution.VmList
2. TaskList=Optimized_Solution.TaskList
3. while(i< VmList.length)

$$\text{LoadList} = \text{TaskList.size} * \text{cloudletsgroupsonvms\_len} / \text{VmMips}_i$$

4. $\text{Cap\_VM}_i = \text{PE}_{\text{Num}} * \text{PE}_{\text{MIPS}} + \text{Vm}_{\text{BW}}$
5. End While
6. while(i< VmList.length)
7. $\text{TimeTaken\_VM}_i = \text{LoadList}_i / \text{Cap\_VM}_i$

$$\text{Deviation} = \sqrt{\frac{1}{\text{No\_Tasks}} \sum_{i=0}^{N} (\text{Time\_VM}_i - \overline{\text{ATimeVM}})^2}$$

8. If (deviation>ATimeVM)
9. SYSTEM IS IMBALANCED
10. Flag=false
11. Else
12. SYSTEM IS BALANCED
13. Flag=true
14. End IF
15. Deviation_VM=(double)sd_vms.get(j);
    a. if(Deviation_VM >ATimeVM)
    b. OverloadedVM
    c. Overloaded_flag=true
    d. else
    e. UnderloadedVM
    f. Underloaded_flag=true
16. Endif
17. Sort OverloadedVM and UnderLoadedVM
18. Clusters=MakeClusters(TaskList)
19. While(TaskList!=0)
20. If (overloaded_flag=true)
    a. Choose the task assigned to overloaded machines and assign to the low loaded machines
    b. sendNow(underloadedVM, CloudSimTags.CLOUDLET_SUBMIT,TaskCluster1);
21. Else
    a. Choose the task assigned to underloaded machines and assign to the more loaded machines
    b. sendNow(OverloadedVM, CloudSimTags.CLOUDLET_SUBMIT,TaskCluster2);
22. EndIF

Algorithm 4 Procedure for Honey Bee Load Balancer

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2023/223310 Volume 10, Issue 4, July – August (2023)

RESEARCH ARTICLE

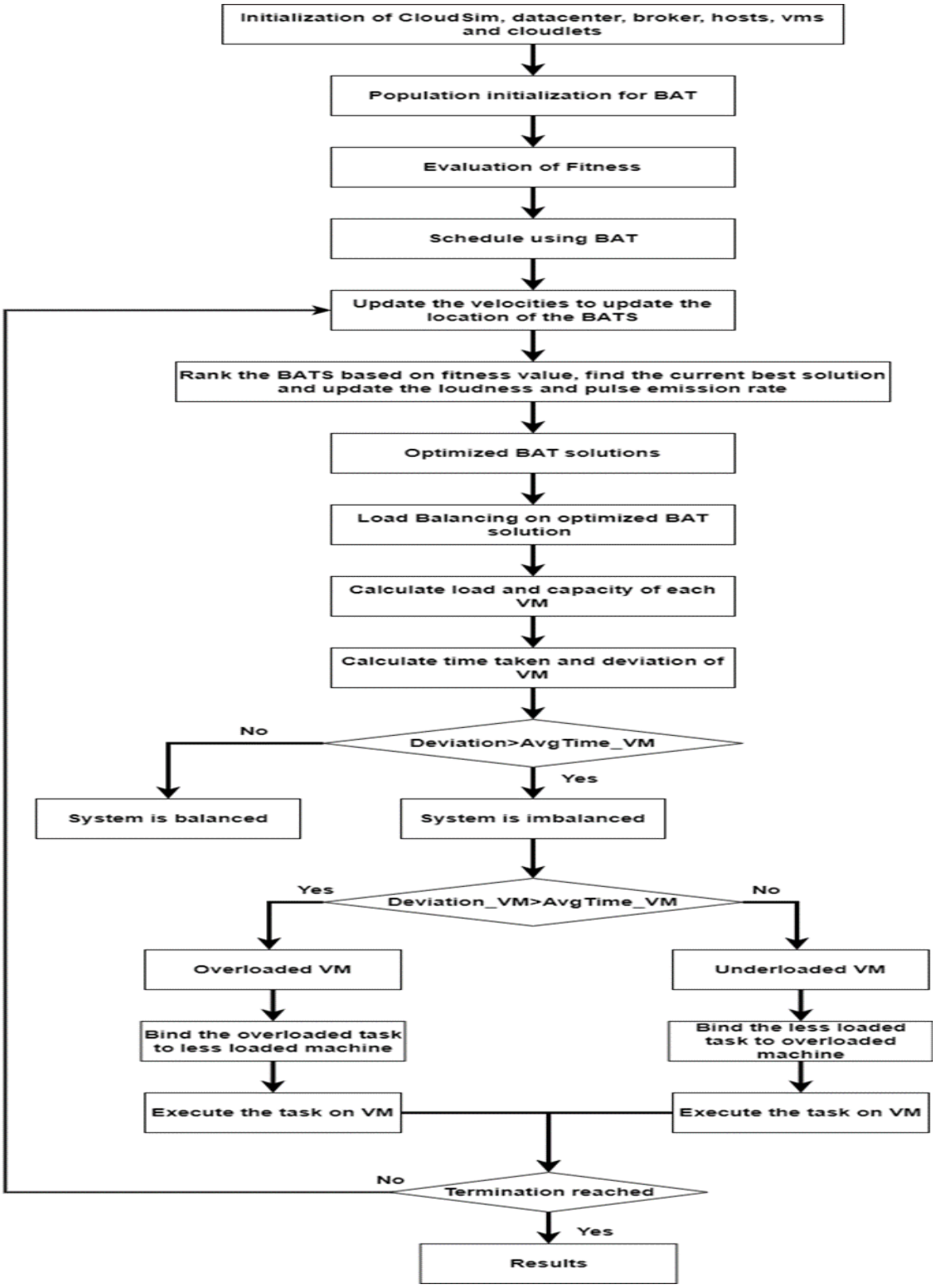

Figure 6 Flowchart of Proposed Algorithm

RESEARCH ARTICLE

In this research, we proposed Honey Bee based improvised BAT algorithm for load balancing method. These characteristics serve two key purposes for the algorithm. Set priorities to prevent allocating overloaded virtual machines. Each VM's load, which mostly depends on the command duration of the work, is taken into consideration while determining the priority of each job's allocation to the appropriate VM. The approach achieves load balancing across virtual machines by limiting more assignments to overworked virtual machines whose processing times vary above a threshold. The components of cloud computing are a collection of data centers, each of which is made up of a collection of m hosts and j virtual machines. Work distribution is the algorithm's subsequent phase. The population size is randomly and uniformly distributed across various VMs. Where i stands for the population and j for the current task, Bat represents the solution as given by Bat with position Xij2=[0…..n] and velocity Xij. The bat must now find prey more quickly in the dimension it inhabits after coming up with a novel solution. The update amount and pulse emission rate are formed by uniformly distributed random numbers that are generated. Data should be moved to the overloaded virtual machines as soon as it is known that some virtual machines are being overloaded by several processes. These characteristics serve two key purposes for the algorithm. Set priorities to prevent allocating overloaded virtual machines. Each VM's load, which mostly depends on the command duration of the work, is taken into consideration while determining the priority of each job's allocation to the appropriate VM. The approach achieves load balancing across virtual machines by limiting more assignments to overworked virtual machines whose processing times vary above a threshold.

We need to gather data since the last allocation and deallocation in order to identify which jobs to assign to which virtual machines. If there is honey in the garden, the procedure is identical to which sources the bees should visit. There are two recognized categories of information. The first type is threshold data, which gauges host availability in response to host changes. The virtual machine level does the same availability tests. Second: Priority information with up-to-date load data. Processing time at the host level and job count at the virtual machine level are two examples of load metrics. The first job is taken out of the queue once the job locates a list of available hosts. To obtain another job from the waiting queue, send the control flag. This activity might be accepted by several hosts. Therefore, depending on the priority information, the task should identify the best host. The host with the lowest load should be taken into account. The presence of a host indicates the availability of one or more virtual machines. The list of virtual machines that are accessible on a given host is then determined by rules at this level. The chosen host may have numerous virtual machines that can take the requested job. Examine the virtual machine's workload, the host and virtual machine's combined processing speed, and the hosts' and virtual machines' respective overload conditions. Give each virtual machine you find a task.

5. RESULTS AND DISCUSSIONS

This section discusses the analyzed results and assesses the performance of optimization algorithms, including Ant Colony Optimization (ACO), BAT algorithm, Particle Swarm Optimization (PSO), ACO Load, PSO Load, and the (proposed) BAT Load (Honey Bee-based BAT Load) algorithm in a cloud computing environment. Table 1 provides the simulation parameters used in the study.

Table 1 Simulation Parameters

Cloudsim Objects	Input Parameters	Value
Task(Cloudlet)	Len_Task	1000-5000
	#Tasks	2400-4000
Virtual Machine	#Vms	50
	MIPSVMs	350-1000
	RAMVMs	1024-4048
	BWVMs	100-1200
	Scheduler_Jobs	Time shared and Space shared
	#Pes	1-3
Datacenter	#Datacenter	10
	# Host	2-10
	Scheduler_VM	Time shared and Space shared

The proposed method was executed with conventional algorithms Ant Colony Optimization, BAT algorithm, Particle Swarm Optimization, and the same algorithm with Honey Bee Load balancer in the identical dataset by the results. We determined the degree of imbalance, the execution time, the overall processing time, and the completion time for this investigation.

5.1. Optimizing Test Set

To improve scheduling in the cloud network, both the conventional BAT algorithm and the newly proposed BAT Load algorithm are applied. Assume that there are 100 people in the population, 100 extreme iterations, 0.5 burst firing rate, and frequencies of 1 and 0 for the maximum and minimum. Each algorithm runs through 100 iterations based on the values for the aforementioned parameters.

5.2. Makespan Time

It is the time it takes for all tasks in a sequence to complete, or the time it takes for the last task to complete. Furthermore, it's

RESEARCH ARTICLE

obvious to see that as there are more clouds, the proposed BAT Load's standard deviation gets lower and smaller. Concerning other algorithms, the proposed BAT Load has the lowest standard deviation. With an increase in Cloudlet, its value falls to 0, showing that the data divergence and deviation are the least and that the time taken is also the shortest. Equation (1) can be denoted as

$$MST = \max\{CT_{kl} | k \in T.k = 1,2 \ldots . . n \; and \; l \in VM.l = 1,2, \ldots . . m\} \quad (1)$$

Table 2 shows the results after running each algorithm for 100 iterations for the various number of cloudlets varying from 100 to 500, the makespan time of the proposed BAT Load algorithm is better in all the cases as compared to other traditional ACO, PSO, and BAT algorithms and ACO and PSO with Honey Bee algorithm. It can also be seen that for 500 cloudlets, the makespan time for the proposed BAT Load algorithm is 109.64 ms, and ACO and PSO with Honey Bee Algorithm is 119.87 ms and 178.73 ms.

Table 2 Comparison on the Basis of Makespan

Makespan Time (in ms)						
No of Tasks	ACO	PSO	ACO Load	PSO Load	BAT	Proposed BAT Load
100	35.63	58.03	21.99	35.71	33.27	18.46
200	76.05	113.95	41.46	46.02	71.2	35.08
300	108.49	147.46	64.55	95.96	103.74	56.31
400	138.93	202.16	89.26	169.84	129.84	83.36
500	197.6	228.93	119.87	178.73	143.05	109.64

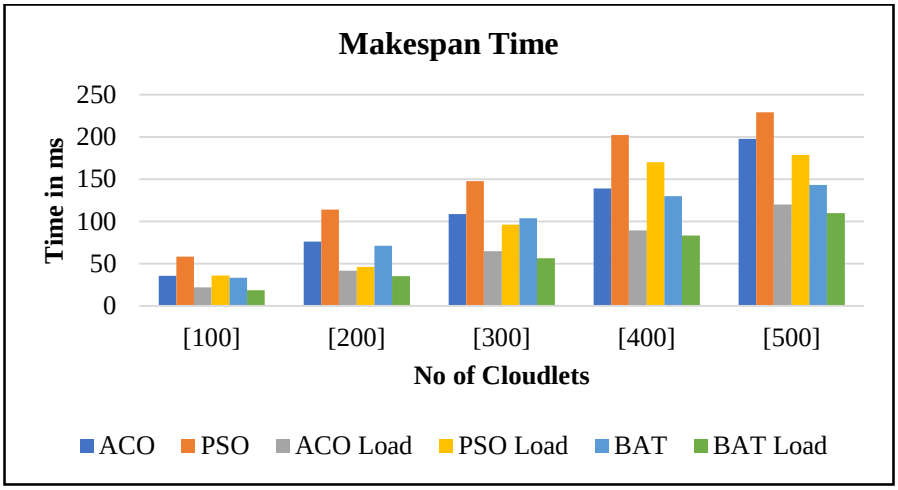

Figure 7 Comparison of Makespan Time

Figure 7 shows the make-span time results against the cloudlets, for the profound algorithm and the other five existing methods. It is clear that, in comparison to the other ways, the profound method predicted a less makespan time, being followed by the BAT, ACO Load, ACO, PSO Load, and PSO algorithm. It is substantially lower than the Proposed BAT Load method, particularly with Cloudlet at 500. A very little amount of variation exists across all methods for the cloud values of 100 and 500 according to the normal distribution. The proposed BAT Load makespan, which is the best and longest among the values of 300 and 400, is also the longest.

5.3. Processing Time

Time allotted to complete the task by CloudSim clock. According to the testing findings depicted in Figure 8, the proposed BAT Load's processing time is unquestionably shorter than that of other algorithms when compared to the quantity of Cloudlets. ACO and PSO initially handle data quite similarly, but when the number of clouds rises, ACO's

RESEARCH ARTICLE

processing time outpaces PSO significantly. It is analyzed by the following equation (2) and (3):

$$\text{Processing}_{\text{time}} = \text{cloudlet}_{\text{length}}/(\text{vm}_{\text{MIPS}} * \text{no}_{\text{ofPES}}) \quad (2)$$

$$\text{Total Processing}_{\text{time}} = \sum_{i=1}^{n} \text{cloudlet length}_i/(\text{vm}_{\text{MIPS}} * \text{no}_{\text{ofPES}}) \quad (3)$$

Figure 6 and Table 3 illustrate the comparison of processing time performance parameters of proposed and other algorithms. For 500 tasks and 50 virtual machines, the total processing time of BAT Load is 2106.13 ms, and ACO and PSO with Honey Bee Algorithm is 1905.23 ms and 2571.204 ms.

Figure 8 shows the total processing time versus the number of cloudlets for ACO, PSO, BAT, ACO Load, PSO Load and the (proposed) BAT Load algorithms, thus, the profound method, which is next to the BAT algorithm in terms of overall processing time, is superior to other methods. As the number of Cloudlets increases, ACO has a tendency to allocate a large number of Cloudlets to virtual security resource nodes with higher performance, which results in an unusually long processing time overall. Processing times tend to increase as PSO and ACO Load decline to local maxima in subsequent phases. As the number of Cloudlets rises, BAT Load dramatically cuts down on processing time. To maintain a lot of Cloudlets in a safe cloud, loading BAT is the best option.

Table 3 Comparison on the Basis of Total Processing Time

Total Processing Time(in ms)						
No of Tasks	ACO	PSO	ACO Load	PSO Load	BAT	Proposed BAT Load
100	765.821	1395.91	418.57	1180.966	338.01	338.9
200	1075.209	1856.824	875.87	1461.302	723.35	718.6
300	1237.652	2372.283	1076.31	1849.255	1150.69	1139.41
400	1686.367	2502.71	1419.2	2250.081	1615.79	1602.47
500	2390.18	2973.78	1905.23	2571.204	2120.19	2106.13

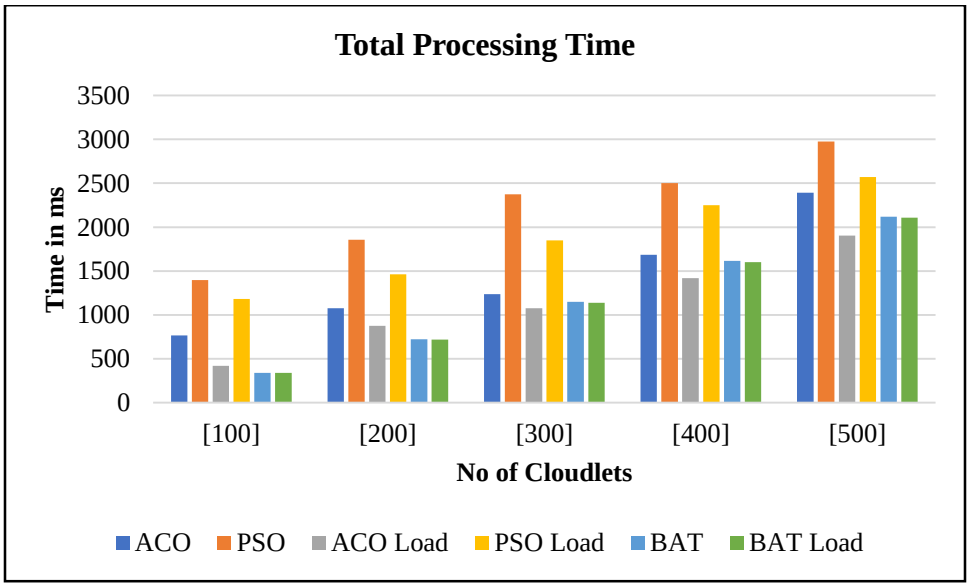

Figure 8 Comparison of Total Processing Time

5.4. Processing Cost

It is the cost of processing to complete a specific task using a process. Given that the computation involves costs, the lower the outcome, the better. Table 4 shows that as the quantity and size of uniformly and normally distributed clouds rise, so do the costs associated with all techniques. According to the standard deviation, the proposed BAT Load tends to have a lesser standard deviation than those of ACO, PSO Load, and

RESEARCH ARTICLE

ACO Load while having a bigger standard deviation than BAT. The fact that it has the lowest BAT cost and the greatest ACO cost is also obvious. It can be denoted as following Equation (4) and (5):

$$Processing_{cost} = DataCenter_{costpermemory} * VM_{RAM} \quad (4)$$

$$Total\ Processing_{cost} = \sum_{i=1}^{k} DataCenter_{costpermemory} * VM\ RAM_{k} \quad (5)$$

Where k is number of virtual machines

Better algorithms are those with lower costs. As the number of clouds rises, Table 4 demonstrates how the cost of all algorithms climbs exponentially. The proposed BAT load has the lowest cost and PSO has the highest, according to the analysis of the table 4.

The presented approach outperformed the other algorithms in terms of total processing cost and was followed by the BAT algorithm, as shown in Figure 9. The total processing cost is plotted against the number of cloudlets. Figure 9 shows that the (proposed) BAT Load is superior to ACO and PSO, but inferior to ACO and PSO Load regardless of uniform or normal distribution. Additionally, ACO has the highest level of unpredictability, cost input, and variance and variability about Cloudlets. In other words, resource utilization is not favorable to improvement, and energy consumption is at its maximum.

Table 4 Comparison on the Basis of Total Processing Cost

Total Processing Cost (in $)						
No of Tasks	ACO	PSO	ACO Load	PSO Load	BAT	Proposed BAT Load
100	9312.065	9951.704	3368.842	4223.649	9949.36	3357.12
200	17706.13	18469.03	6892.491	8566.96	19331.72	6604.13
300	22427.76	25045.85	10166.53	11625.512	28692.98	9752.73
400	26301.69	32384.62	13325.75	17534.112	37944.13	12767.78
500	31433.8	38054.33	16370.18	24292.76	47516.26	15782.83

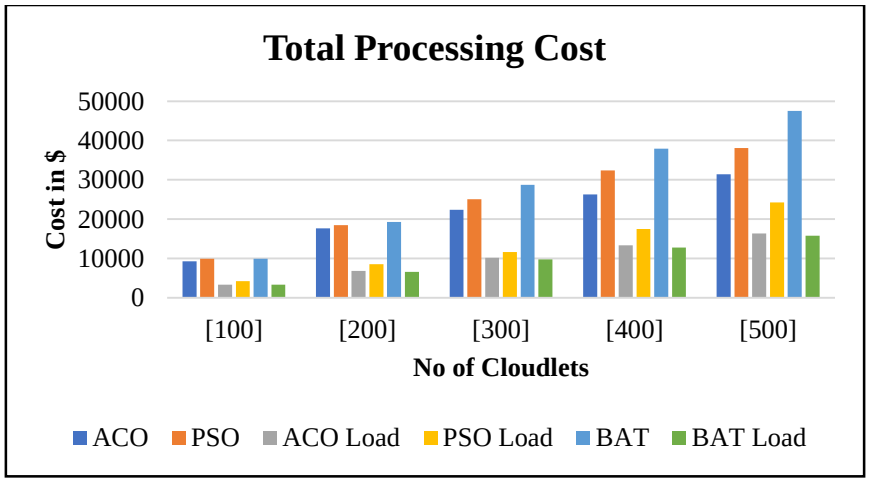

Figure 9 Comparison of Total Processing Cost

5.5. Execution Time

The time required to complete a task. The degree of change between the end point (task) and the execution start point (task). Equation (6) can be expressed as follows.

$$Execution\ Time = FinishTime\ of\ Task - ExecStartTime\ of\ Task \quad (6)$$

Table 5 show the execution time of the five methods as a function of the number of cloudlets.

Figure 10 show the execution time of the five methods as a function of the number of cloudlets. According to Figure 10 and Table 5 the proposed approach produced the best outcomes for this metric. When the BAT algorithm was combined with Honey Bee Algorithm more optimal results

RESEARCH ARTICLE

are achieved as compared with the other algorithms. As a result, the strategy suggested in the Results offers an improvement over other existing ways in terms of producing superior results. We assess how various Cloudlet-proposed methods perform and take longer to execute than existing approaches. Following an analysis of the data, we contrasted the findings with those of other recently developed methods like ACO loading and PSO loading. To create findings that were superior to those of other methods now in use, we improved the method for producing the results.

Table 5 Comparison of Total Execution Time by Varying Number of Tasks

Total Execution Time (in ms)						
No of Tasks	ACO	PSO	ACO Load	PSO Load	BAT	Proposed BAT Load
100	854.41	1191.82	828.39	962.22	676.03	671.23
200	1787.6	3313.64	1711.44	1872.61	1446.69	1410.48
300	2818.82	5144.56	2660.95	2872.92	2301.38	2217.84
400	3943.18	7605.41	3673.51	4851.53	3231.59	3094.47
500	5095.03	9347.62	4749.6	6710.66	4240.38	4037.71

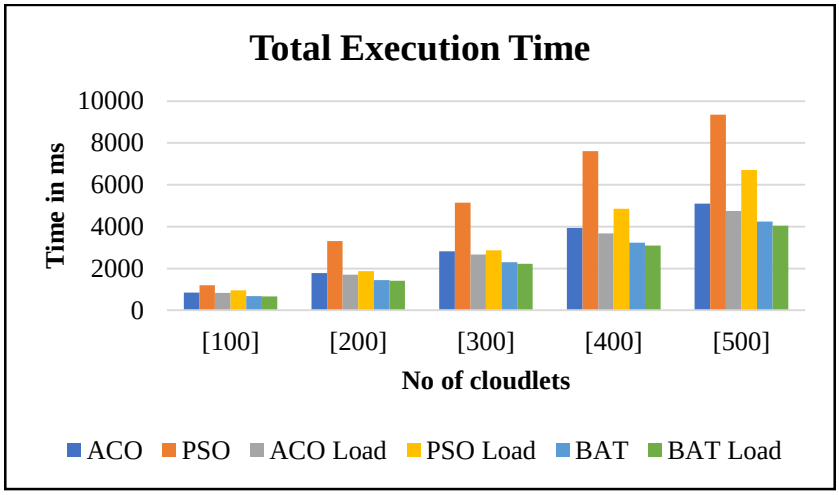

Figure 10 Comparison of Execution Time

5.6. Degree of Imbalance

Introduces a level of imbalance to gauge an unbalanced load on a virtual computer. This function will show the percentage of each VM's running capacity that is currently loaded. An acceptable level of imbalance should be taken into account while resource scheduling in order to improve performance. Likewise, the standard deviation and both the average and ideal values of the proposed BAT load imbalance are nearly identical. The proposed BAT Load algorithm beats other metaheuristic algorithms when taking the degree of imbalance into account because its standard deviation still falls within the lowest range when compared to other algorithms, demonstrating its significant qualities. This is a possible way to translate formula (7).

$$D_i = (T_{max} - T_{min}) / T_{avg} \quad (7)$$

where T_{max} and T_{min} are minimum & maximum T_i among all VMs, T_{avg} is average T_i of VMs. Figure 11 and Table 6 illustrate the performance of the degree of imbalance analysis. It can be analyzed that the (proposed) BAT Load is superior in this metric, followed by BAT, ACO Load, ACO, PSO Load, and PSO. For 500 cloudlets, the degree of imbalance for BAT load is 177.08, whereas for ACO load and PSO load is 197.86 and 202.40. Overall, the performance of BAT load is the best among all five mentioned algorithms. In contrast to a normal distribution, a uniform distribution exhibits less fluctuation. Analysis shows that the PSO imbalance is most pronounced in a regularly distributed setting. The optimum BAT load imbalance is also comparable to BAT load outcomes.

RESEARCH ARTICLE

Table 6 Comparison on the Basis of Degree of Imbalance

Degree of Imbalance						
No of Tasks	ACO	PSO	ACO Load	PSO Load	BAT	Proposed BAT Load
100	40.39	41.25	40.24	41.15	39.52	36.28
200	80.91	79.19	79.65	73.49	79.38	71.08
300	120.69	120.18	118.83	117.51	120.4	106.5
400	160.06	164.06	159.91	158.49	159.77	141.84
500	201.62	211.27	197.86	202.4	200.99	177.08

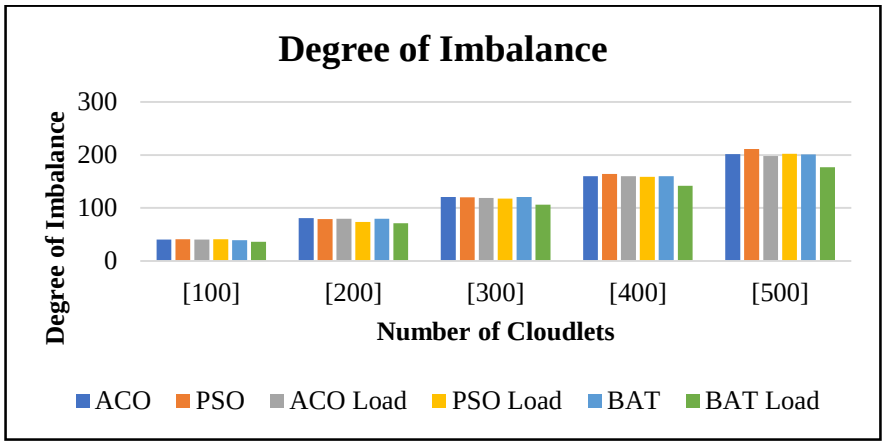

Figure 11 Comparison of Degree of Imbalance

5.7. Discussion

This paper was motivated by the desire to enhance virtual machine task distribution mechanisms and lower idle time in cloud computing in dynamic contexts. As a result, proposed Honey Bee based improvised BAT algorithm design in this study to enhance the task allocation strategy in cloud computing virtual machines. The suggested approach may divide the load across each virtual machine, enhance resource utilization, and considerably shorten the makespan time. Through 100 tests, the amount of tasks, task size, VMs in various areas, and DCs are constantly changed in order to replicate the algorithms performance. To simulate situations, uses the CloudSim simulation program. The results show that each data center's execution of the proposal required some time. Load has the lowest standard deviation of the BAT. This value decreases to zero as the number of Cloudlets rises, suggesting low time and data skew. The proposed BAT load algorithm performs better for substantially assigned activities and data sizes when compared to other conventional algorithms. The suggested cloud data center host prediction method into practice in the following paper.

6. CONCLUSION

Due to the benefits, it offers, particularly the capabilities of the software and hardware and the equipment's relatively low cost from the user's perspective, cloud computing is growing in popularity. This study provides a Bat with Honey Bee load balancing approach to address the load balancing issue in task scheduling, and it also discusses and evaluates the effectiveness of various performance metrics. In order to address and optimize resource and load balancing of scheduling issues in cloud computing, it compares and evaluates the effectiveness of several optimization algorithms, including ACO, PSO, Bat, and ACO and PSO with the honey bee load balancer. Making the best use of the resources at hand, the research's objective is to investigate the scheduling problem in the context of cloud computing. As a consequence, the Bat Load suggested in this study efficiently uses resources by balancing make span, degree of imbalance, cost, execution time, and processing time. Graphs and statistical analysis are used to interpret the results which demonstrates that the Bat Load algorithm performs more effectively than the existing optimization method at balancing

RESEARCH ARTICLE

load and allocating resources in the cloud computing environment.

REFERENCES

[1] Arya, Lokesh Kumar, and Amandeep Verma. "Workflow scheduling algorithms in cloud environment-A survey." Recent Advances in Engineering and Computational Sciences (RAECS) (2014): 1-4.

[2] Chiang, Mao-Lun, Hui-Ching Hsieh, Wen-Chung Tsai, and Ming-Ching Ke. "An improved task scheduling and load balancing algorithm under the heterogeneous cloud computing network." In 8th International Conference on Awareness Science and Technology (iCAST), pp. 290-295. IEEE, 2017.

[3] Sharieh, Ahmad, and Layla Albdour. "A heuristic approach for service allocation in cloud computing." International Journal of Cloud Applications and Computing (IJCAC), 7(4), (2017): 60-74.

[4] S. Sefati, M. Mousavinasab, and R. Zareh Farkhady. "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability." Journal of Supercomputing, 78 (2022), pp. 18-42.

[5] Houssein, Essam H., Ahmed G. Gad, Yaser M. Wazery, and Ponnuthurai Nagaratnam Suganthan. "Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends." Swarm and Evolutionary Computation, 62 (2021): 100841.

[6] Milan, Sara Tabaghchi, Lila Rajabion, Hamideh Ranjbar, and Nima Jafari Navimipour. "Nature inspired meta-heuristic algorithms for solving the load-balancing problem in cloud environments." Computers & Operations Research, 110 (2019): 159-187.

[7] Akilandeswari, P., and H. Srimathi. "Survey and analysis on task scheduling in cloud environment." Indian Journal of Science and Technology, 9, no. 37 (2016): 1-6.

[8] Masdari, Mohammad, Sima ValiKardan, Zahra Shahi, and Sonay Imani Azar. "Towards workflow scheduling in cloud computing: a comprehensive analysis." Journal of Network and Computer Applications, 66 (2016): 64-82.

[9] Sharma, Shabnam, Ashish Kr Luhach, and S. A. Sinha. "An optimal load balancing technique for cloud computing environment using bat algorithm." Indian Journal of Science Technology 9, no. 28 (2016): 1-4.

[10] R. Kumari, and A. Jain. "An efficient resource utilization based integrated task scheduling algorithm." in: 4th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2017, pp. 519–523.

[11] S. Rani, and P. Suri,. "An efficient and scalable hybrid task scheduling approach for cloud environment." International Journal of Information Technology, (2018) 1–7.

[12] A.A. Nasr , N.A. El-Bahnasawy , G. Attiya and A. El-Sayed. "Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint." Arabian Journal for Science and Engineering, 44 (4) (2019), 3765–3780

[13] Xiang, B.; Zhang, B.; and Zhang, L. "Greedy-ant: ant colony system inspired workflow scheduling for heterogeneous computing." IEEE Access 5, 11404–11412 (2017)

[14] M.R. Thanka , P.U. Maheswari and E.B. Edwin. "An improved efficient: artificial bee colony algorithm for security and QoS aware scheduling in cloud computing environment." Journal of Cluster Computing, 22 (5) (2019) 10905–10913 .

[15] Kruekaew, Boonhatai, and Warangkhana Kimpan. "Enhancing of artificial bee colony algorithm for virtual machine scheduling and load balancing problem in cloud computing." International Journal of Computational Intelligence Systems, 13, no. 1 (2020): 496-510.

[16] Xing, Huanlai, Fuhong Song, Lianshan Yan, and Wei Pan. "A modified artificial bee colony algorithm for load balancing in network-coding-based multicast." Soft Computing, 23, no. 15 (2019): 6287-6305.

[17] Rani, Preeti, Prem Narayan Singh, Sonia Verma, Nasir Ali, Prashant Kumar Shukla, and Musah Alhassan. "An Implementation of Modified Blowfish Technique with Honey Bee Behavior Optimization for Load Balancing in Cloud System Environment." Wireless Communications and Mobile Computing 2022(5) (2022).

[18] Jeyalaksshmi, S., J. Anita Smiles, D. Akila, Dibyendu Mukherjee, and Ahmed J. Obaid. "Energy-Efficient Load Balancing Technique to optimize Average response time and Data Center Processing Time in Cloud Computing Environment." In Journal of Physics: Conference Series, vol. 1963, no. 1, p. 012145. IOP Publishing, 2021.

[19] Babu L.D Dhinesh, and Krishna Venkata P., "Honey bee behavior inspired load balancing of tasks in cloud computing environments." Applied Soft Computing 13(5), 2013, 2292-2303.

[20] D. Chaudhary , B. Kumar and R. Khanna. " Npso based cost optimization for load scheduling in cloud computing," in: International Symposium on Security in Computing and Communication, Springer, 2017, pp. 109–121 .

[21] J.A.J. Sujana , T. Revathi , T.S. Priya , and K. Muneeswaran. "Smart PSO-based secured scheduling approaches for scientific workflows in cloud computing,." Journal of Soft Computing, 23 (5) (2019) 1745–1765 .

[22] P. Guo and Z. Xue. "Cost-effective fault-tolerant scheduling algorithm for real-time tasks in cloud systems", in: 2017 IEEE 17th International Conference on Communication Technology (ICCT), IEEE, 2017, pp. 1942–1946

[23] Zheng, Jianguo, and Yilin Wang. "A hybrid multi-objective bat algorithm for solving cloud computing resource scheduling problems." Sustainability 13, no. 14 (2021): 7933.

[24] Raj, Gaurav, Shabnam Sharma, and Aditya Prakash. "Modified Bat Algorithm for Balancing Load of Optimal Virtual Machines in Cloud Computing Environment." In Applications of Artificial Intelligence and Machine Learning, pp. 475-488. Springer, Singapore, 2022.

[25] Barzegar, Behnam, Samaneh Habibian, and Mehrnoush Fazlollah Nejad. "Heuristic algorithms for task scheduling in Cloud Computing using Combined Particle Swarm Optimization and Bat Algorithms." Journal of Advances in Computer Research, 10, no. 3 (2019): 83-95.

[26] Gu, Yi, and Chandu Budati. "Energy-aware workflow scheduling and optimization in clouds using bat algorithm." Future Generation Computer Systems, 113 (2020): 106-112.

[27] Adhikari, Mainak, Sudarshan Nandy, and Tarachand Amgoth. "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud." Journal of Network and Computer Applications, 128 (2019): 64-77.

[28] Ibrahim, Laheeb Mohammed, and Ibrahim Ahmed Saleh. "A solution of loading balance in cloud computing using optimization of bat swarm algorithm." Journal of Engineering Science and Technology, 15, no. 3 (2020): 2062-2076.

[29] Choi H, Ahn N, and Park S. "An ant colony optimization approach for the maximum independent set problem." Journal of the Korean Institute of Industrial Engineers, 33(4) (2007):447–456.

[30] Arora T, and Gigras Y. "A survey of comparison between various meta-heuristic techniques for path planning problem", International Journal of Computer Engineering & Science, vol.3 (2013):62–66.

Authors

Mr. Abhishek Gupta is currently working as an Assistant Professor in CSE department, GBPIET (An Autonomous Institute of Govt. of Uttarakhand) Pauri-Garhwal, Uttarakhand, India. He did his B.Tech from UPTU Lucknow in 2010 and M.Tech from JUIT Solan in 2012. He is having 10+ years of teaching experience. His areas of interest include computer networks, distributed systems, cloud computing, machine learning etc.

RESEARCH ARTICLE

Dr. H S Bhadauria is currently working as Professor in CSE department, GBPIET (An Autonomous Institute of Govt. of Uttarakhand) Pauri-Garhwal, Uttarakhand, India. He did his PhD from IIT Roorkee. He is having 25+ years of teaching experience. His areas of interest include computer networks, cloud computing, machine learning, image processing etc.

How to cite this article:

Abhishek Gupta, H.S. Bhadauria, "Honey Bee Based Improvised BAT Algorithm for Cloud Task Scheduling", International Journal of Computer Networks and Applications (IJCNA), 10(4), PP: 494-510, 2023, DOI: 10.22247/ijcna/2023/223310.