Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

OPTIMIZING CLOUD PERFORMANCE: A COMPARATIVE ANALYSIS OF BIRD SWARM AND ANT COLONY ALGORITHMS FOR LOAD BALANCING

YOGITA YASHVEER RAGHAV *
K R Mangalam University, Gurugram, Haryana. *Corresponding Author Email: ygtraghav@gmail.com

PALLAVI PANDEY
IILM University, Gurugram, Haryana.

Abstract

Load balancing plays a vital role in the realm of cloud computing by efficiently dispersing workloads across multiple servers or resources, which serves to enhance overall performance, availability, and scalability. The primary goal of load balancing is to optimize resource utilization and prevent server overload, thereby optimizing the entire cloud infrastructure. In addressing the challenges associated with workload distribution and resource utilization optimization in the cloud, researchers have devised algorithms inspired by natural processes likeevolution, swarm behavior, and genetics. This research assesses the performance of two such algorithms, namely ant colony optimization (ACO) and bird swarm optimization (BSO), with a focus on load balancing. A comparative analysis is carried out using various parameters, including fitness score, throughput, resource utilization, and makespan. The findings demonstrate that the BSO algorithm surpasses the ACO algorithm in terms of fitness score, throughput, resource utilization, and makespan. To conduct these experiments, the CloudSim simulator is utilized within the NetBeans development environment.

Keyword: ACO, BSO, Nature Inspired Algorithms, Makespan, Throughput.

1. INTRODUCTION

Cloud computing has brought about a revolution in accessing and utilizing computing resources for individuals and businesses. Through cloud technologies, users can avail themselves of diverse resources like databases, storage, networking, infrastructure, platforms, and applications via the internet. The key advantage of cloud computing lies in relieving users from the burden ofphysically managing the underlying infrastructure. IaaS allows for the provisioning of virtualized computing resources, including networking, storage, and virtual machines. PaaS empowers users to create, manage, and deploy applications on a platform without worrying about the underlying infrastructure. SaaS enables users to access software programs such as email, CRM, or ERP without requiring local installation or updates. Figure 1 depicts the available cloud computing services. Cloud computing presents numerous advantages, including scalability, flexibility, and cost-effectiveness. Cloud computing enables users to easily adjust their computing resources according to changing needs, eliminating the requirement for extra investments in hardware or infrastructure. Additionally, the pay-as-you-go model ensures users only pay for the resources they use, resulting in cost savings. This transformative technology has revolutionized the accessibility and utilization of computing resources for both individuals and organizations. Its significant impact is projected to continue driving digital transformation in organizations for the foreseeable future [1].

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

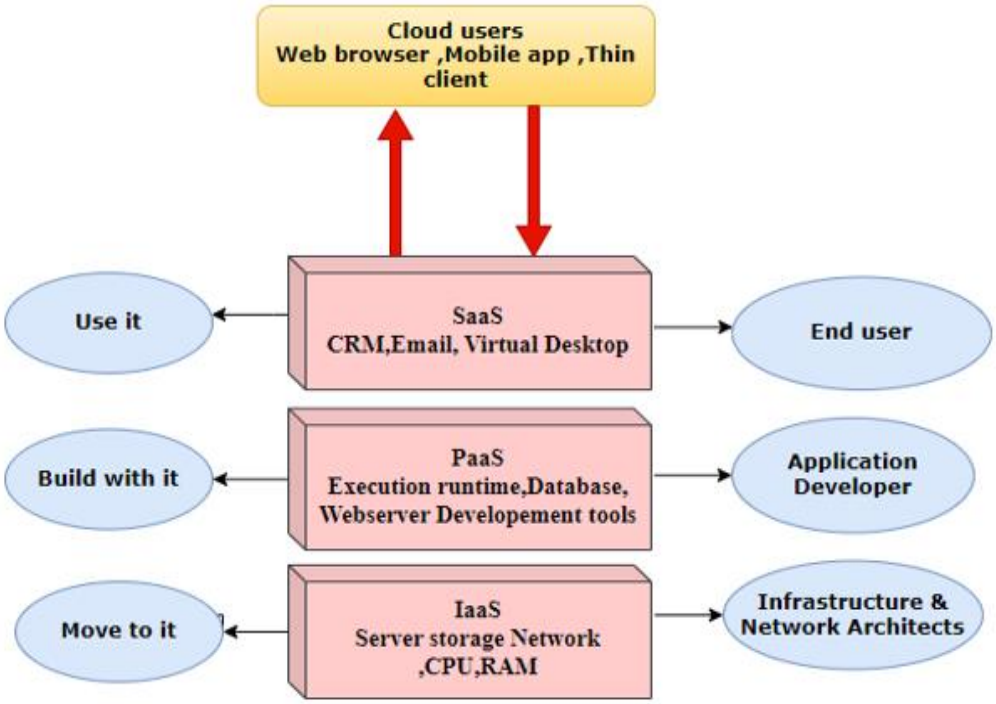

Figure 1: Cloud computing services

1.1 Load balancing in the cloud computing

Load balancing is the technique to disseminate the workloads across computing resources in cloud environments. By using load balancing incoming traffic can be disseminated among multiple servers to satisfy the frequently changing workload demand. It enhances the performance and preserves the continuous services. It also enables the distribution of workloads among multiple geographic regions. In a typical scenario, a load balancer acts as an intermediary between client devices and multiple backend servers or applications, distributing incoming traffic among them according to a predefined algorithm or set of rules. Load balancing helps to prevent any single server or application from becoming overwhelmed with too much traffic, thus avoiding service disruptions and downtime [12].

2. LITERATURE STUDY

The stability of processing multiple jobs in a cloud environment must be maintained, though, and this is a challenging problem. Therefore, it needs a load-balancing method that distributes the task to the VMs without impacting the system's performance. The literature on cloud computing environments uses a variety of load balancing approaches. This section lists the benefits and drawbacks of the existing works done in load balancing in cloud environment. Literature study with different parameters has been elaborated in table 1.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Table 1: Comparison of various algorithms through a study

Reference	Author (s)	Technique	Evaluation	Compared with	Findings	Simulator used
[2]	Kaushik Mishra et a	Binary BSO-inspired load balancing technique using a binary variant.	Achieve improved makespan and utilization by implementing a well-suited fitness function.	Round Robin	There was a 22% improvement in resource utilization, coupled with a 33% reduction in makespan.	Cloudsim
[3]	Pradhan, A et al	Efficient task scheduling on cloud resources was achieved through the implementation of the LBMPSO algorithm.	Achieves optimal efficiency by Minimizing makespan And Maximizing Resource utilization.	PSO	The proposed algorithm surpasses existing techniques in Reducing makespan and Enhancing resource utilization.	Cloudsim
[4]	Fatemeh et al.	Non-preemptive scheduling with PSO Based approach	Balanced system Reduces response time, improves resource utilization and performance	Round Robin (RR) Task scheduling,	Utilization. Improved Resource utilization by 22% and Reduced makespan by 33% compared to basic PSO.	Cloudsim
[5]	Talha Akhtar et al.	Particle utilized PSO and GSOCK algorithms	Improved efficiency	PSO	GSO surpassed PSO and CK exhibiting significant performance enhancements of 71.17%, 74.14% and 84.15% in networks featuring 50,100 and 200 nodes during peak load.	Clousim
[6]	B. Mallikar juna, P. Venkata Krishna	Bee Colony	Degree imbalance, makespan, Task migration	FCFS and Dynamic load balancing algorithm	The BSO Algorithm outperformed FCFS and DLB in reducing makespan, as verified through the iteration process assessing VM overload	Cloudsim

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

[7]	G.Shobana et al.	honey bee	Optimized resource utilization for Faster response times.	Number of tasks And duration of task	Preemptive Cloudsim Task scheduling mimics honey bees' foraging behavior to allocate VMs, optimizing makespan, throughput, and datacenter performance.	Cloudsim
[8]	Hinesh Babu et al.	Inspired by Honey bee behaviour	Execution time and waiting time	WRR, FIFO, DL B: Loa d balancing techniques Round Robin and Fuzzy GSO.	Proposed algorithm balances task priorities, minimizing waiting time effectively on machines.	Cloudsim
[9]	Shabnam Sharma et al.	BAT algorithm	Job migration and response time	Round Robin and Fuzzy GSO.	Virtual machine job migration impacts response time during load balancing.	Parallel Processing toolbox" in Mat lab.
[10]	Gund ipika Kaur	Adaptive firefly algorithm (ADF)	Response time, processing Time	ACO	ADF algorithm outperformed ACO by reducing response time for users and datacenter processing through Parameter comparisons.	Cloudsim
[11]	Kethavath Prem Kumar	Firefly Algorithm with Cuckoo Search	Migration Time	HBB-LB, DLB, HDLB and CMLB	The CS-FA Method migrated a mere two tasks, whereas the HDLB method migrated seven tasks when 40 loads were taken into account.	Cloudsim

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

3. ANT COLONY OPTIMIZATION AND BIRD SWARM OPTIMIZATION

Ant Colony Optimization (ACO) and Bird Swarm Optimization (BSO) are metaheuristic optimization techniques employed for load balancing in cloud computing. ACO emulates ant foraging behavior, while BSO mimics the collective movement of bird flocks in search of food, offering effective strategies for achieving optimal resource allocation and task distribution in cloud environments. Both algorithms aim to distribute the workload among available resources while optimizing the allocation of resources to meet demand. They expand the search space and adapt to changes in workload and resource usage.

3.1 Proposed algorithm using Ant colony optimization

ACO has been successfully applied to solve load balancing problems in cloud computing. The algorithm uses a pheromone matrix to store information about the quality of solutions and simulates the foraging behavior of ants, with each ant representing a job orrequest. The pheromone level and heuristic data guide the movement of the ants from one server to another, and the pheromone level is updated based on the quality of the ants' discoveries. The equations used in ACO for load balancing can be divided into pheromone updating and ant decision-making. The objective is to ensure that no server is overloaded while minimizing response times and maximizing resource utilization. [13], [14]. The equations used in ACO for load balancing in cloud computing can be categorized into two main parts: pheromone updating and ant decision-making.

Pheromone updating equation

$$\tau ij = (1 - \rho)\tau ij + \Delta\tau ij \qquad (1)$$

Here, τij represents the pheromone level of the edge between servers i and j, ρ is the evaporation rate, and Δ τij is the amount of pheromone deposited by the ants that traverse the edge.

Ant decision-making equation

$$Pij = \tau ij\alpha \times \eta ij\beta / \Sigma (\tau ik\alpha \times \eta ik\beta) \qquad (2)$$

In the equation, the probability of selecting server j from server i is indicated as Pij. The weighting factors for pheromone level and heuristic information are α and β, respectively. The heuristic information for the edge connecting server i and j is denoted as ηij [15].

These equations are repeatedly applied to identify the best allocation of virtual machines to servers that optimizes resource utilization and minimizes response time. As the ants traverse the search space and update the pheromone matrix, the algorithm converges to a solution.

3.2 Proposed algorithm using Ant colony optimization

Step 1: Initializing ant positions involves randomly assigning N ants to servers and allocating tasks in a randomized manner across the servers.

Step 2: Initialize and assign threshold to each node.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Step 3: for each ant

begin for

VM = {VM1, VM2, VM3, . . . , VMm} where VMj (j ∈ {1, 2, . . . , m}) represents the number of VM on which task Ti (i ∈ {1, 2, . . . , n}) is going to be processed. Create pheromone table using equation 1 & 2. Calculate Completion time (TTC), make span, Response time and, Average resource utilization using equation 10,11,12,14.

end for

Step4: for each

Estimate the load using equation 6, 7, 8.

end for

Step 5: Find the state of the vm group using equation 18

While(ULVM!=NULL)

begin loop

Calculate used resource using equation 16 and available resource using equation 17.

End while

Step 6: if CPU utilization >threshold

Then node is overloaded

Step 7: for each ants

Calculate fitness value using equation 13.

Set the current fitness value as the new best position.

If the best overloaded node is chosen, then use equation 19 to determine how long each VM will take to migrate from the overloaded node else using equations 1, 2, update the position of the ants.

else

find next optimal solution; end if

end for

Step 8: Select VM with minimum migration time;

Step 9: While (OLVM!=null and ULVM!=null) begin loop

Get Task list which need to transfer from selected Overloaded VM;

end while

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Step 10: While (ULVM! =null)

Sort the ants for searching the underloaded node using the routing table; Path construction using equation 2;

Update pheromone table & routing table using equation 1

if all ants complete their tour then

Calculate fitness values using equation 13.

end if

for each ants,

if Best under loaded node selected then delegate the task to the perfect node;

else

find next optimal solution; end if

end while

Step 11: Update current utilization of CPU

3.2.1 Illustration of Proposed algorithm using ACO by flowchart

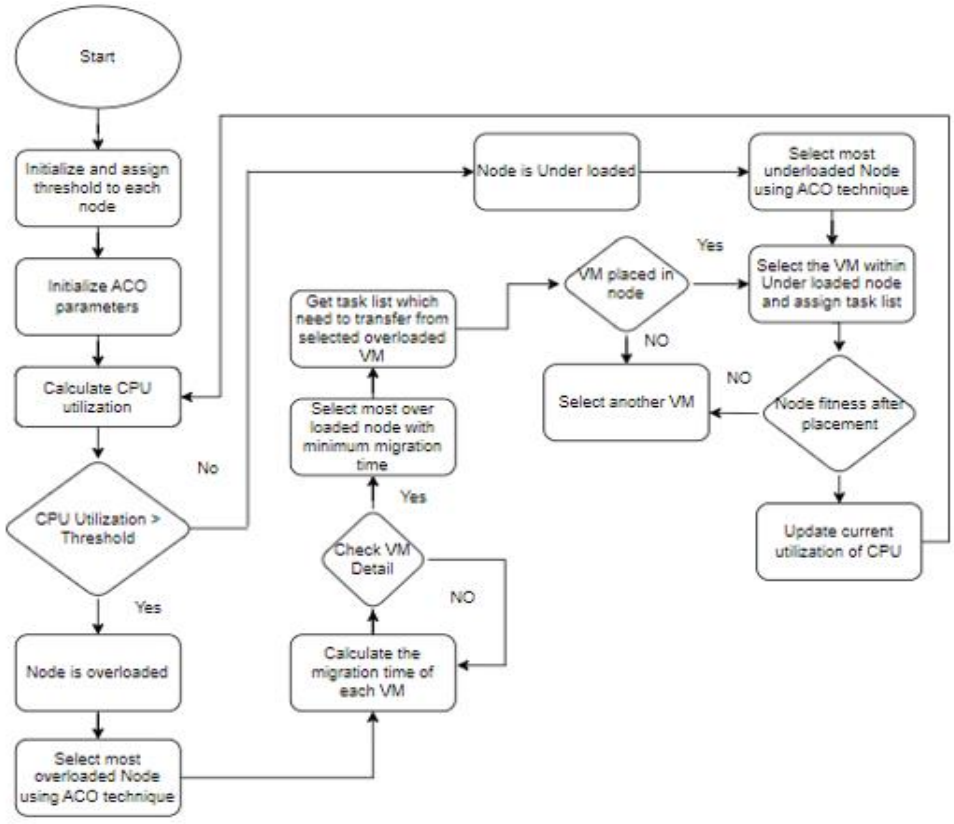

Figure 2: Flowchart of proposed algorithm using ACO

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

3.2.2 Proposed algorithm using Bird swarm optimization

Inspired by the collective behavior of birds, Bird Swarm Optimization (BSO) is a metaheuristic algorithm that has found application in diverse optimization problems, such as load balancing in cloud computing. Utilizing Bird Swarm Optimization (BSO), the allocation of virtual machines (VMs) to physical servers can be optimized. The goal is to minimize response time for requests, maximize resource usage, and avoid server overload simultaneously. The BSO algorithm works by simulating the flocking behavior of birds, where each bird represents a solution candidate. The algorithm uses three key factors: Through separation, alignment, and cohesion, the birds are led to the best course of action. The concept of cohesion ensures that the birds maintain their group structure, alignment ensures that they move in the same direction, and separation ensures that they maintain a safe distance from each other. The BSO equations employed in cloud computing has two categories such as: bird movement and bird decision-making. [16].

Bird movement equation

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1) \qquad (3)$$

where $x_{ij}(t)$ is the position of bird i in dimension j at time t, $v_{ij}(t+1)$ is the velocity of bird i in dimension j at time t+1, and $v_{ij}(t+1)$ is calculated using the separation, alignment, and cohesion factors [17-18].

Bird decision-making equation

$$f(x_{ij}) = c1 \times f1(x_{ij}) + c2 \times f2(x_{ij}) \qquad (4)$$

where $f(x_{ij})$ is the fitness function of the solution candidate, c1 and c2 are the weighting factors of the separation and cohesion factors, respectively, and $f1(x_{ij})$ and $f2(x_{ij})$ are the fitness functions based on the distance between the VMs and the physical servers.

These equations are employed iteratively to iteratively search for an optimal allocation of VMs to servers, aiming to minimize response time and maximize resource utilization. As the birds move within the search space, adjusting their velocities based on separation, alignment, and cohesion factors, the algorithm gradually converges to a solution [19-20].

3.2.2 Proposed algorithm using BSO

Step1: Define birds and initialize bird's position and initialize the necessary parameters and Pheromone trails.

Step2: Initialize and assign threshold to each node.

Step3: for each birds

> VM = {VM1, VM2, VM3, . . . , VMm} where VMj ($j \in \{1, 2, \ldots, m\}$) represents the number of VM on which task Ti ($i \in \{1, 2, \ldots, n\}$) is going to be processed. Calculate Completion time (TTC), make span, Response time and, Average resource utilization using equation 10,11,12,14.

end for

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Step4: for each

Estimate the load using equation 6, 7, 8. end for

Step5: Find the state of the vm group using equation 18 While(ULVM!=null)

begin loop

Calculate used resource using equation 16 and available resource using equation 17.

End while

Step6: if CPU utilization >threshold Then node is overloaded

Step 7: for each birds,

Calculate fitness value using equation 13.

Set the current fitness value as the new best position.

If the best overloaded node is chosen, then

use equation 19 to determine how long each VM will take to migrate from the overloaded node

else

end if end for

using equations 3, 4, update the position of the birds. find next optimal solution;

Select VM with minimum migration time;

Step 8: While (OLVM!=null and ULVM!=null)

Get Task list which need to transfer from selected Overloaded VM; end while

Step 9: While (ULVM! =null)

Calculate used resource using equation 16 and available resource using equation 17.

End while

Step6: if CPU utilization <threshold Then node is underloaded

Step 7: for each birds,

Calculate fitness value using equation 13.

Set the current fitness value as the new best position. if Best under loaded node selected then

delegate the task to the perfect node;

else

find next optimal solution; end if

end for

Step10: Update current utilization of CPU

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Flowchart of proposed BSO load balancing algorithm

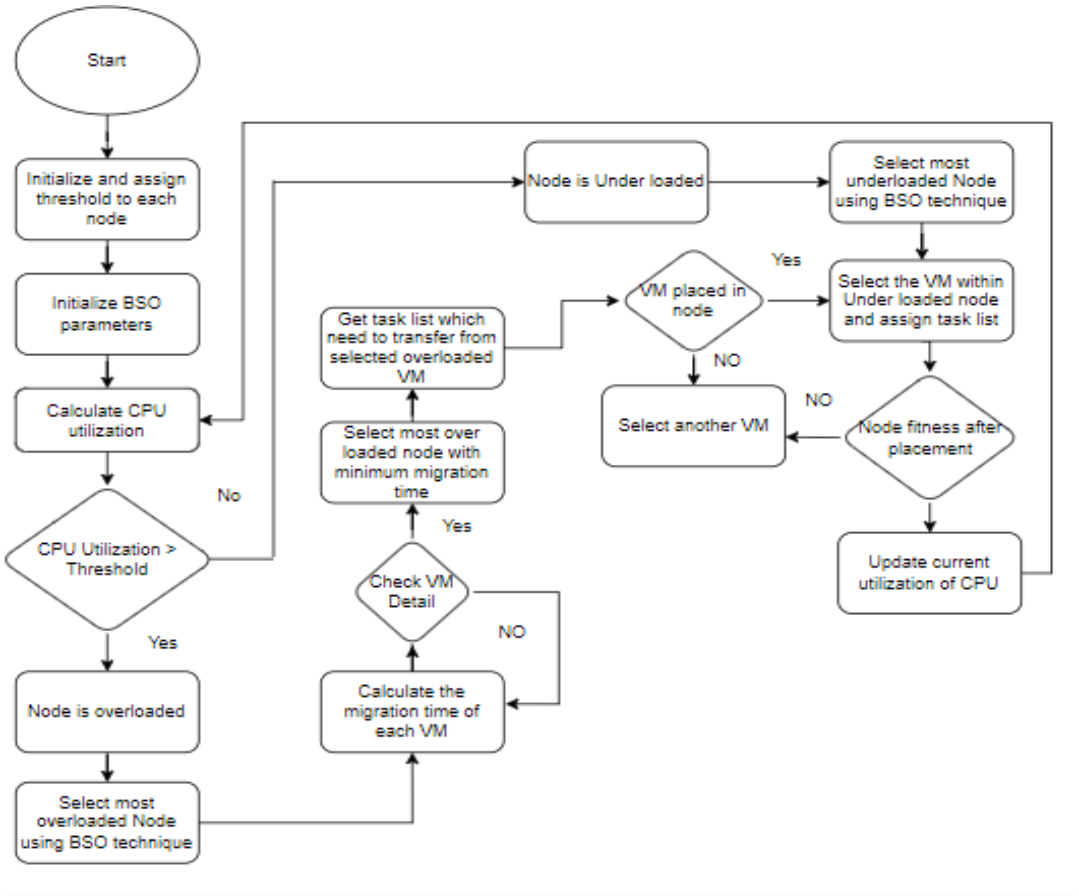

Figure 3: Flowchart of proposed algorithm using BSO

3.2.3 Proposed load balancing problem formulations

The proposed load balancing method in this study is motivated by ant colony optimization and bird swarm optimization. There is a correlation between distributing loads and foraging in swarms. In the context of the cloud,each bird in the swarm sounds as a particle. Similar to how birds search for food, the tasks are divided among the VMs. Overloaded VM behavior is exhibited by empty or already explored food sources. Therefore, it is necessary to locatea new underload node on which to migrate the tasks of overloaded nodes. In order to determine which particle is in the best position, the fitness values of each particle are compared using the fitness function that has been set for the particular problem. The following definitions are used to define the load balancing problem [21].

1. Task set: Let a set of task , Task ={ Taskx1 ,Taskx2,Taskx3.....Taskxn } (5)

where Taski , 1<=i<=xn , is ith task with set of instructions.

2. VM Set: Let a set of VM = { Vm1,Vm2,Vm3,.....Vmn } (6)

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

where VMj, 1<=i<=mn is deployed under the physical machine or host.

3. VM Load: Load (VMi, t) = (($Task$,t)) / (SR (VM ,t)) (7)

where t is the task, NT is the number of tasks and SR is the service rate. VM Load represents the load of a specific virtual machine (VMi) at a given time (t). It is calculated by dividing the number of tasks (NT) assigned to the VM by its service rate (SR). This equation helps assess the workload on individual VMs based on the number of tasks and their processing capabilities [22].

4. Load: = ∑ (i=1) ^m [($Load$ (VMi, t)] (8)

Where m is the number of VM and t is the task. Load represents the total load in the system. It is calculated by summing the individual loads of m VMs, denoted as Load (VMi, t), where i ranges from 1 to m (the total number of VMs). The load of each VM is determined based on the specific task (t) assigned to it. This equation allows for aggregating the loads of multiple VMs to assess the overall system load.

5. Host Load: Host Load= [(1/m)] ∑ (i=1) ^m (Load (VMi, t))" (9)

This equation represents the average load on a host. It calculates the host load by summing the loads of all virtual machines (VMs) running on the host at a given time (t) and dividing it by the total number of VMs (m). It provides a measure of the overall workload distribution across the host.

6. Completion Time (CT): task completion refers to the process of completing a given workload or task that has been assigned to a particular resource (such as a server).

CT (i j) = Finish time (Taskj)- Start time(Taskj) (10)

7. Makespan (MS): This metric demonstrates how long it takes to complete all jobs that are submitted to the system within a certain time unit. It is the overall amount of time needed to do all jobs that have been sent to the system. The greatest amount of time the host needs to run through the data centre is called the system's makespan. The proper system load balancing is the outcome of the ideal makespan [1].

MS = max {Task CT (i j) | i=1, 2, 3, 4... n; j=1, 2, 3, 4....., m} (11)

8. Response time (RT): Response time in load balancing refers to the amount of time it takes for a host to respond to a request from a client. In the context of load balancing, response time refers to the amount of time it takes for a load balancer to receive a request and forward it to a host, and for the host to process the request and send a response back to the load balancer [23].

RT = n * Task CT (ij) (12)

9. Fitness score (Fval): In the present study, fitness value has been calculated based on the makespan (completion time) and virtual machine utilization. Following equation has been used for finding out fitness value.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

$$Fval = \frac{1}{Makespan} * VMUtilization \quad (13)$$

The makespan represents the total time taken to complete a set of tasks or jobs in a scheduling problem. It is typically measured as the time elapsed from the start of the first task to the completion of the last task. A shorter makespan indicates better performance. Virtual Machine Utilization refers to the extent to which virtual machines (VMs) are utilized or occupied by tasks or workloads. It can be measured as the ratio of the total time that VMs are processing tasks to the total time available for processing. Higher utilization implies better utilization of resources. The fitness value formula combines these two components by taking the inverse of the product of the makespan and virtual machine utilization. This means that as the makespan decreases or the virtual machine utilization increases, the fitness value will increase, indicating a better solution.

10. VM utilization: The number of resources (including memory, CPU, and network bandwidth) that a virtual machine is using at any particular time is known as VM (Virtual Machine) utilization. In a cloud computing environment, virtual machines are used to host applications and services, and monitoring their utilization is important for ensuring that they are running efficiently and effectively [24,31]. VM Utilization: It represents the overall utilization of virtual machines in the load balancing scenario.

$$\text{VM Utilization} = \frac{\sum_{i=1}^{n}}{makespan * m} \quad (14)$$

Here i refers to individual VM instances in the load balancing setup. The summation symbol suggests that you iterate over all VM n: It represents the total number of VM instances in the load balancing setup instances. Makespan: The makespan is the total time taken to complete a set of tasks or jobs in a system, m represents the average utilization of a VM instance or server.

11. Throughput: Throughput typically refers to the rate at which Cloudlets (representing tasks or workloads) are processed or completed by the simulated cloud infrastructure within a given time period. It is a measure of the system's ability to efficiently handle and process tasks, which can include tasks related to data processing, computation, or other cloud-related operations.

Throughput = Number of Completed Cloudlets / Simulation Time (in seconds) (15)

Higher throughput values indicate that your simulated cloud infrastructure is handling a larger number of tasks efficiently within the given time frame.

12. Consumed Resources ($\vec{T}$_Rused): $\vec{T}$_Rused represents the overall amount of resources consumed by all the tasks in the system. It is computed by summing up the resource vectors of each individual task ($\vec{T}$_i) from i = 1 to n.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

$$\vec{T}\_Rused = \sum\_(i=1)^n \vec{T}\_i. \quad (16)$$

13. Available Resources ($\vec{T}$_Ravail): $\vec{T}$_Ravail represents the resources that are still accessible or available to the underloaded VMs in the system. It is calculated by subtracting the consumed resources ($\vec{T}$_Rused) from the total resources of the underutilized VMs ($\vec{T}$_R).

$$\vec{T}\_Ravail = \vec{T}\_R - \vec{T}\_Rused. \quad (17)$$

By using these equations, you can determine the consumed resources by aggregating the resource vectors of all the tasks, and then calculate the available resources by subtracting the consumed resources from the total resources of the underutilized VMs. These computations help in understanding the resource utilization and availability in the system, which can be further utilized for load balancing or resource allocation decisions [25,30].

14. State of the node: Every node's load will be compared to a threshold value to determine its condition; if it exceeds the threshold value, the node is overloaded; otherwise, it is under loaded.

If Load>threshold then overloaded node

else (18)

Underloaded node

End if

15. Migration time (MT): The time it takes for a workload or job to be transferred from one resource (such a server) to another is referred to as migration time. In the current study, the virtual machine is being moved from an overloaded node to an under loaded node. The migration time of a VM in CloudSim depends on factors such as VM size, network bandwidth, data transfer rate, and VM state. Larger VMs, higher bandwidth, and faster transfer rates reduce migration time. To calculate the migration time of each virtual machine, following equation can be utilized:

Migration Time = Data Size / Data Transfer Rate (19)

Where Migration Time refers estimated time taken for the VM migration, measured in seconds, Data Size represents size of the VM's disk image or memory that needs to be transferred, typically measured in bytes and Data Transfer Rate refers the estimated rate at which data can be transferred between the source and destination nodes, usually measured in bytes per second [26-29].

4. SYSTEM CONFIGURATION IN CLOUD ENVIRONMENT

In Table 2, 3, and 4 system configurations has been shown such as Datacenter, host and virtual machine configuration.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Table 2: Datacentre Configuration

Attribute's Name	Value
No. of data centres	3
Architecture	x86
OS	Ubuntu
VMM	Xen
Time Zone	10.0
Process Cost	3.0
Memory Cost	0.05
Storage Cost	0.001
Bandwidth Cost	0.1

Table 3: Host Configuration

Attribute's Name	Value
Storage Number of Host	100000MB 1
Host_MIPS	1000
Host RAM	2048MB
Bandwidth of Host	100000Gbps
Number of Host	1

Table 4: Virtual Machine Configuration

Attribute's Name	Value
No. of VMS	40
Image Size	10000MB
RAM	512MB
MIPS	250
Bandwidth	1000Gbps
VMM_NAME	Xen.
VM_PES	1

4.1 Result and Discussion

The present research employed the CloudSim using Netbeans to compare ACO and BSO algorithms. Various metrics such as Best Fitness Score, Throughput, Resource Utilization, and Makespan is evaluated using both algorithms, and the results is analyzed.

Based on Figures 8, it is evident that in case of Makespan, BSO surpasses ACO. According to Figure 8, ACO yields a Makespan of approximately 10.804 seconds, while BSO achieves a significantly lower Makespan of about 1.8787 seconds for the same number of tasks. These findings strongly suggest that employing the BSO algorithm can lead to optimal Makespan results.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

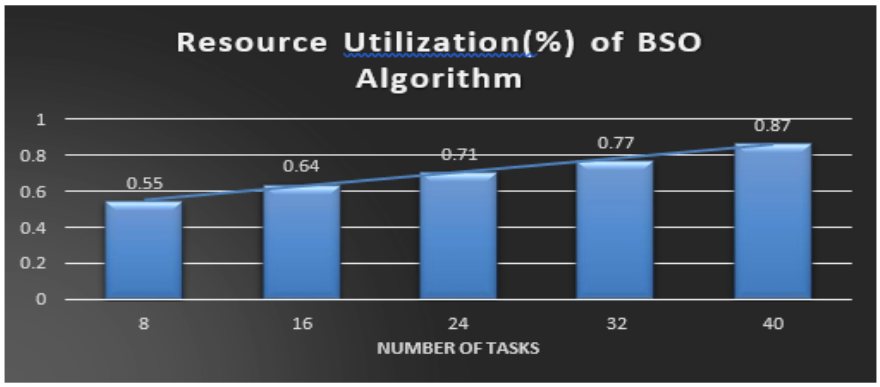

Figure 4: Resource utilization of Proposed BSO algorithm

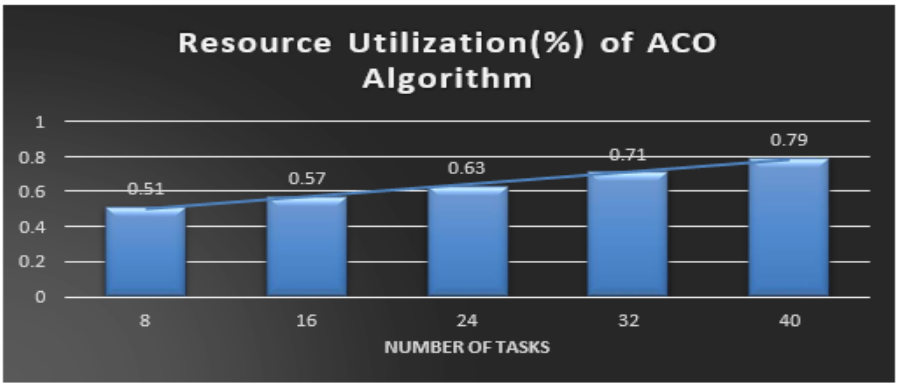

Figure 5: Resource utilization of Proposed ACO algorithm

Figure 4 and 5 demonstrates that BSO utilise resources more efficiently than ACO. Figure 5 shows that the resource utilisation of ACO is[0.51,0.57, 0.63,0.71,0.79] when there are 8, 16, 24, 32, and 40 tasks, while Figure 4 shows that the resource utilisation of BSO is [0.55, 0.64,0.71,0.77,0.87]% for the same number of tasks. Based on our evaluation, we can conclude that the BSO algorithm is capable of achieving optimal resource utilization, making it an effective approach to enhance overall performance.

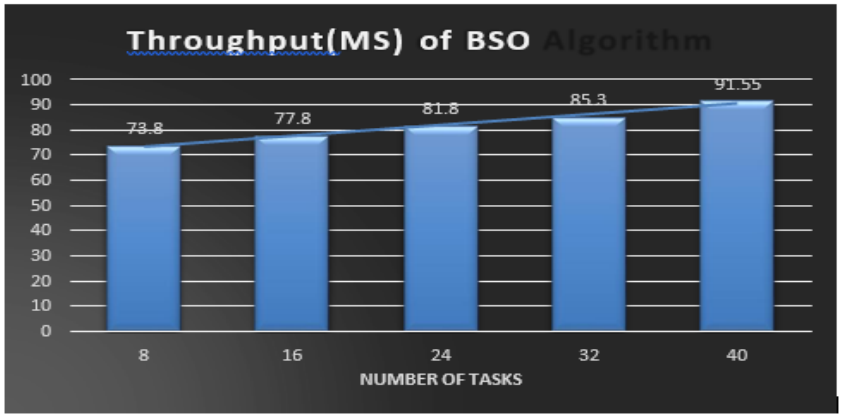

Figure 6: Throughput of Proposed BSO algorithm

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

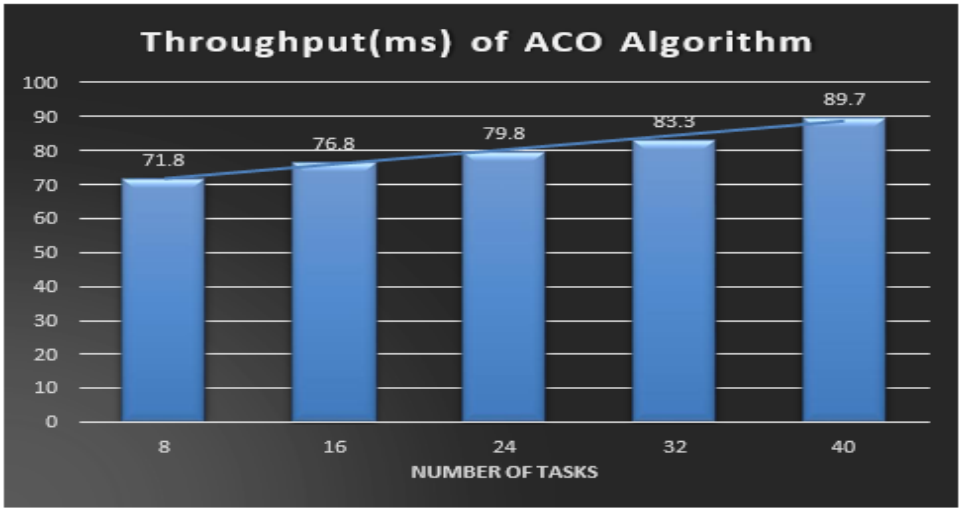

Figure 7: Throughput of Proposed ACO algorithm

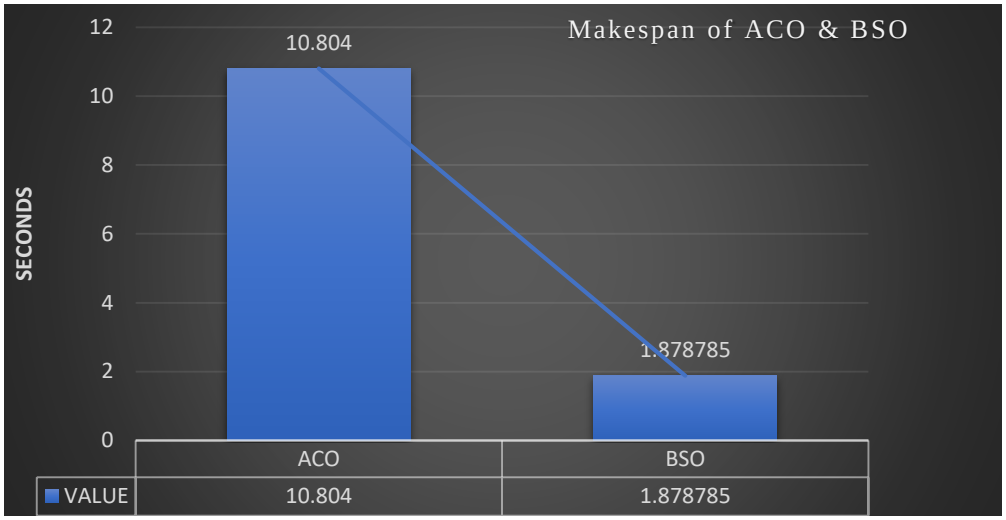

Figure 8: Makespan of ACO and BSO algorithm

Matrices	ACO	BSO
Makespan	10.804 seconds	1.878785 seconds
Throughput	[71.8,76.8, 79.8,83.3,89.7]%	[73.8,77.8,81.8,85.3,91.55]%
Resource Utilization	[0.51,0.57, 0.63,0.71,0.79]	[0.55,0.64,0.71,0.77,0.87]

Figure 9: Comparison of Results of ACO and BSO both

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

Figure 6 and 7 demonstrates that BSO provide more throughput than ACO. Figure 7 shows that the throughput of ACO is [71.8,76.8, 79.8,83.3,89.7] % when there are 8, 16, 24, 32, and 40 tasks, while Figure 6 shows that the throughput of BSO is [73.8 ,77.8 ,81.8 ,85.3 ,91.55] % for the same number of tasks. Results concludes that the BSO algorithm is capable of achieving optimal throughput, thereby enhancing performance.

Summary and Future Directions

For load balancing in cloud computing systems, ACO and BSO are efficient metaheuristic optimization techniques. The difficulty of the problem could, however, affect how well they do. BSO is preferable for issues with smaller search space and fewer variables, whereas ACO is appropriate for issues with a complicated search space and many variables. Regarding makespan, throughput, fitness score, and resource use, BSO performed admirably in this investigation. Both algorithms demonstrate the ability to address diverse optimization problems and draw inspiration from the cooperative behaviour observed in social animals. To further improve their performance, a hybrid algorithm that combines the strengths of ACO and BSO could be developed. For example, a hybrid algorithm could integrate ACO's pheromone trail updating mechanism with BSO's individual repulsion and social attraction rules to create a new algorithm that outperforms both individual algorithms.

Conflicts of Interest: No conflicts of interest.

References

1) Raghav, Y. Y., & Vyas, V. (2019, October). A comparative analysis of different load balancing algorithms on differentparameters in cloud computing. In 2019 3rd International Conference on Recent Developments in Control, Automation & Power Engineering (RDCAPE) (pp. 628-634). IEEE

2) Mishra, K., & Majhi, S. K. (2021). A binary Bird Swarm Optimization based load balancing algorithm for cloud computing environment. Open Computer Science, 11(1), 146-160.

3) Pradhan, A., & Bisoy, S. K. (2022). A novel load balancing technique for cloud computing platform based on PSO. Journal of King Saud University-Computer and Information Sciences, 34(7), 3988-3995.

4) Ebadifard, F., & Babamir, S. M. (2018). A PSO-based task scheduling algorithm improved using a load-balancing technique for the cloud computing environment. Concurrency and Computation: Practice and Experience, 30(12), e4368.

5) Akhtar, T., Haider, N. G., & Khan, S. M. (2022). A comparative study of the application of glowworm swarm optimization algorithm with other nature-inspired algorithms in the network load balancing problem. Engineering, Technology & Applied Science Research, 12(4), 8777- 8784.

6) Mallikarjuna, B., & Krishna, P. V. (2018). A nature inspired bee colony optimization model for improving load balancing in cloud computing. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(2S2), 51 -54.

7) Shobana, G., Geetha, M., & Suganthe, R. C. (2014, February). Nature inspired preemptive task scheduling for load balancing in cloud datacenter. In International conference on information communication and embedded systems (ICICES2014) (pp. 1-6). IEEE.

8) LD, D. B., & Krishna, P. V. (2013). Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied soft computing, 13(5), 2292-2303.

ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

9) Sharma, S., Luhach, A. K., & Sinha, S. A. (2016). An optimal load balancing technique for cloud computing environment using bat algorithm. Indian J Sci Technol, 9(28), 1-4.

10) Kaur, G., & Kaur, K. (2017). An adaptive firefly algorithm for load balancing in cloud computing. In Proceedings of Sixth International Conference on Soft Computing for Problem Solving: SocProS 2016, Volume 1 (pp. 63-72). Springer Singapore.

11) Kumar, K. P., Ragunathan, T., Vasumathi, D., & Prasad, P. K. (2020). An efficient load balancing technique based on cuckoo search and firefly algorithm in cloud. Algorithms, 423, 422-432.

12) Raghav, Y. Y., Vyas, V., & Rani, H. (2022). Load balancing using dynamic algorithms for cloud environment: A survey. Materials Today: Proceedings, 69, 349-353.

13) K. Surjeet, P. Sabyasachi, and A. Ranjan, “Turkish Journal of Computer and Mathematics Education Vol. 12 No. 11 (2021), 3885- 3898 Research Article A Particle Swarm and Ant Colony Optimization based Load Balancing and Virtual Machine Scheduling Algorithm for Cloud Computing Environment A Parti,” vol. 12, no. 11, pp. 3885–3898, 2021.

14) Navtej Singh Ghumman, Rajwinder Kaur."Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system" , 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2015.

15) Jeyakrishnan, V., & Sengottuvelan, P. (2017). A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms. Wireless Personal Communications, 94, 2363-2375.

16) Jean Pepe Buanga Mapetu, Zhen Chen, Lingfu Kong. "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing", Applied Intelligence, 2019.

17) V. Arulkumar, N. Bhalaji. "Performance analysis of nature inspired load balancing algorithm in cloud environment", Journal Ambient Intelligence and Humanized Computing, 2020.

18) Sambit Kumar Mishra, Bibhudatta Sahoo, Priti Paramita Parida. "Load balancing in cloud computing: A big picture”, Journal of King Saud University - Computer and Information Sciences, 2020.

19) Navtej Singh Ghumman, Rajwinder Kaur."Dynamic combination of improved max-min and ant colony algorithm for load balancing in cloud system" , 2015 6th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2015.

20) Mishra, R & Jaiswal, A 2012, ‘Ant colony optimization: A solution of load balancing in cloud’, International Journal of Web &Semantic Technology (IJWesT), vol. 3, no. 2, pp. 33-50.

21) Parida, B. R., Rath, A. K., & Mohapatra, H. (2022). Binary self-adaptive salp swarm optimization-based dynamic load balancing in cloud computing. International journal of information technology and web engineering (IJITWE), 17(1), 1-25.

22) Mousavi, S. M., & Gábor, F. (2016). A novel algorithm for Load Balancing using HBA and ACO in Cloud Computing environment. International Journal of Computer Science and Information Security, 14(6), 48.

23) Xue, S., Li, M., Xu, X., Chen, J., & Xue, S. (2014). An ACO-LB Algorithm for Task Scheduling in the Cloud Environment. J.Softw., 9(2), 466-473.

24) Dave, A., Patel, B., & Bhatt, G. (2016, October). Load balancing in cloud computing using optimization techniques: A study. In 2016 International Conference on Communication and Electronics Systems (ICCES) (pp. 1-6). IEEE.

25) Jeyakrishnan, V., & Sengottuvelan, P. (2017). A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms. Wireless Personal Communications, 94, 2363-2375.

Jilin Daxue Xuebao (Gongxueban)/Journal of Jilin University (Engineering and Technology Edition)
ISSN: 1671-5497
E-Publication: Online Open Access
Vol: 43 Issue: 11-2024
DOI: 10.5281/zenodo.14064622

26) Raghav, Y. Y., & Vyas, V. (2023). ACBSO: a hybrid solution for load balancing using ant colony and bird swarm optimization algorithms. International Journal of Information Technology, 1-11.

27) Dewan, M., Mudgal, A., Pandey, P., Raghav, Y. Y., & Gupta, T. (2023). Predicting Pregnancy Complications Using Machine Learning. In D. Satishkumar & P. Maniiarasan (Eds.), Technological Tools for Predicting Pregnancy Complications (pp. 141-160). IGI Global. https://doi.org/10.4018/979-8-3693-1718-1.ch008.

28) Gupta, T., Pandey, P., & Raghav, Y. Y. (2023). Impact of Social Media Platforms on the Consumer Decision-Making Process in the Food and Grocery Industry. In T. Tarnanidis, M. Vlachopoulou, & J. Papathanasiou (Eds.), Influences of Social Media on Consumer Decision-Making Processes in the Food and Grocery Industry (pp. 119-139). IGI Global. https://doi.org/10.4018/978-1-6684-8868-3.ch006.

29) Raghav, Y. Y. & Gulia, S. (2023). The Rise of Artificial Intelligence and Its Implications on Spirituality. In S. Chakraborty (Ed.), Investigating the Impact of AI on Ethics and Spirituality (pp. 165-178). IGI Global. https://doi.org/10.4018/978-1-6684-9196-6.ch011

30) Raghav, Y. Y., & Vyas, V. (2023). A Comparative Analysis Report of Nature-Inspired Algorithms for Load Balancing in Cloud Environment. In Women in Soft Computing (pp. 47-63). Cham: Springer Nature Switzerland.

31) Raghav, Y. Y., & Kait, R. (2024). Edge Computing Empowering Distributed Computing at the Edge. In Emerging Trends in Cloud Computing Analytics, Scalability, and Service Models (pp. 67-83). IGI Global.