Indonesian Journal of Electrical Engineering and Computer Science
Vol. 32, No. 3, December 2023, pp. 1545~1556
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v32.i3.pp1545-1556

Cloud computing: an efficient load balancing and scheduling of task method using a hybrid optimization algorithm

Ravinder Bathini[1,2], NareshVurukonda[3]
[1]Department of Information Technology, Vardhaman College of Engineering, Hyderabad, India
[2]Koneru Lakshmaiah Education Foundation, Green Fields, Vaddeswaram, Guntur, India
[3]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, India

Article Info

Article history:

Received Jul 15, 2022
Revised Mar 20, 2023
Accepted Sep 4, 2023

Keywords:

Balancing approach
Cloud computing
Memory consumption
Resource utilization
Simulation efficiency

ABSTRACT

The cloud computing trend on the internet is vital as it allows data and applications to be managed over the internet instead of requiring personal devices. The job of users is scheduled in the resources of the cloud in order to improve performance. Scheduling tasks is an non-deterministic polynomial (NP)-hard problem, as it may have multiple solutions. Various researchers have proposed different load balancing and job scheduling algorithms to optimize the scheduling process in cloud environments, each with disadvantages. Therefore, this research proposes a novel hybrid load balancing and scheduling of tasks by the whale optimization algorithm (WOA) and seagull optimization algorithm (SOA) in the cloud. This hybrid proposed whale-seagull optimization algorithm (WSOA) optimizes task scheduling in the cloud by reducing processing time, response, and execution time, maximizing central processing unit (CPU) utilization, memory utilization, throughput, reliability, and balancing the load. The algorithm is simulated using the CloudSim toolkit package. As compared with existing approaches, simulation results showed better performance in terms of response time, processing time, execution time, CPU utilization, memory utilization, throughput, and reliability and is analyzed by comparing with the harries hawks optimization (HHO), hybrid dragonfly and firefly algorithm (ADA), spider monkey algorithm (SMA) and bird swarm optimization (BSO).

Corresponding Author:

Naresh Vurukonda
Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation
Vaddeswaram, Guntur, Andhra Pradesh, India
Email: nareshvurukonda789@gmail.com

1. INTRODUCTION

Research on cloud computing has recently gained traction and has been extensively employed in scientific research, education, manufacturing, and telecommunications. In addition to providing recording services, backup, secure data storage, and storage clouds [1], [2] are very convenient for cloud users. Cloud-based educational resources [3], [4] can virtualize multiple types of education hardware to provide seamless access to information for schools, teachers, and students. Using CC, resources such as platforms, software, and hardware are provided on a 'pay-as-you-go' basis. The clients should be charged only for the required resources and offerings without paying for hardware infrastructure. Research is currently focused on task scheduling, green computing, cloud security, resource management, and virtualization. With the rapid growth in cloud computing services, scheduling tasks efficiently to compute resources virtual machines (VMs) based on goals has become increasingly important. In order to maximize the use of resources, minimize user costs, and enhance the performance of CC, scheduling tasks and balancing the load strategies are required [5], [6].

A task scheduler's primary goal is to assign user tasks to appropriate resources, namely VMs or host machines, and to schedule in which order the tasks should be executed within a given resource. Once these tasks have been assigned to these VMs, the tasks are executed on these VMs, which run in parallel, so the load is evenly distributed across the heterogeneous VMs [7]–[9].

Assigning tasks to virtual machines is said to be loaded. Due to the increasing number of diverse activities with varying resource requirements, task load balancing on VMs is essential to scheduling tasks. In load balancing, all nodes (hosts or VMs) are monitored for underloaded and overloaded nodes, and the load is distributed among them accordingly [10]–[13]. With increasing cloud computing services, computing tasks will be submitted to the cloud. To maximize the utilization of resources in the cloud, these tasks should be scheduled for execution to allow every task to be completed on time. This decision-making process is known as "cloud task scheduling," and it profoundly influences the efficiency of the entire cloud infrastructure. Various techniques have been developed to address this issue, for example, min-min, first come first serve (FCFS), max-min and minimum completion time techniques. Despite their adoption in many CC systems, some research has shown that these algorithms are unsuitable for scheduling large amounts of data [14]–[16].

In contrast, metaheuristic algorithms have proved effective and robust for tackling many real-world problems [17]–[19]. For example, Kaur and Kaur [20] suggested a hybrid approach based on ant colony optimization (ACO). In this technique, the ACO was combined with heterogeneous earliest finish time and predict earliest finish time to identify which combination of the technique provided the best results. Another study conducted by Muthsamy and Chandran [21] for task scheduling is based on an artificial bee foraging algorithm. They preempted tasks about different priorities to reduce response and execution times.

Similarly, genetic algorithm-based task scheduling was presented by Pang *et al.* [22] that is combined with the estimation of the distribution algorithm to obtain effective results. To improve the response time to every request of the users, Princess and Radhamani [23] suggested the pigeon-inspired optimization algorithm. Moreover, incoming requests were optimally balanced using the harries hawk's optimization (HHO) algorithm by analyzing overloaded and underloaded VMs. Rani and Suri [24] suggested a hybrid algorithm by merging the gravitational search concept in the ACO algorithm. Using the suggested algorithm, incoming jobs are allocated to the VMs to improve cloud computing's task scheduling by decreasing the makespan. Likewise, an artificial bee and particle swarm algorithm was suggested by Thanka *et al.* [25] for task scheduling. To minimize the makespan, Pradhan and Bisoy [26] presented the modified particle swarm optimization algorithm.

Most of these existing algorithms concentrated on reducing the execution time and response time. They do not consider load balancing factors. To balance resources efficiently, needed intelligent algorithms for resource management. Load-balancing algorithms have always improved planning speed and reduced processing and response times. Therefore, in this research 'a metaheuristic algorithm' is proposed based on a hybridization of the whale optimization algorithm (WOA) and seagull optimization algorithm (SOA) for improving the scheduling of tasks. The proposed algorithm's primary goal is to maximize job allocation efficiency. In brief the following are the contributions of this research:
- To implement the hybrid SOA algorithm and WOA.
- The proposed method is simulated in CloudSim.
- The algorithm reduces the load balancing aspects as: response time, execution time, processing time, and load balancing with an increase in memory utilization, CPU utilization, throughput, and reliability in the cloud applications.

The remainder of the paper is organized as follows. The literature review of cloud scheduling is presented in section 2. In section 3 the proposed technique is depicted and section 4 shows the simulation efficiency of the proposed work. The conclusion of this paper is presented in section 5 and finally, section 6 brings the work to a close with suggestions for future improvements.

2. PROPOSED METHOD

In cloud computing, scheduling is critical for users to finish their tasks on time. As a result, selecting the appropriate optimal resource is crucial, and it falls under the non-deterministic polynomial (NP)-hard issue. Therefore, a hybrid cloud job scheduling algorithm is proposed based on the WOA and the SOA. The following discussion demonstrated the design and pseudocode for the suggested hybrid metaheuristic algorithms. Figure 1 depicts the proposed framework to schedule tasks in the cloud. In cloud infrastructure, they get various jobs that differ in size. The proposed model uses metaheuristic algorithms that aim to complete tasks in the cloud data center. The essential goal of the proposed approach is to allocate jobs to the VMs. Based on the length and runtime of the tasks, the proposed method uses a metaheuristic algorithm to manage their priority. The tasks are organized within the task queue once they have been allocated with their priorities. The hybrid algorithm carries out the task scheduling process, which is based on data center properties such as time, memory, and CPU.

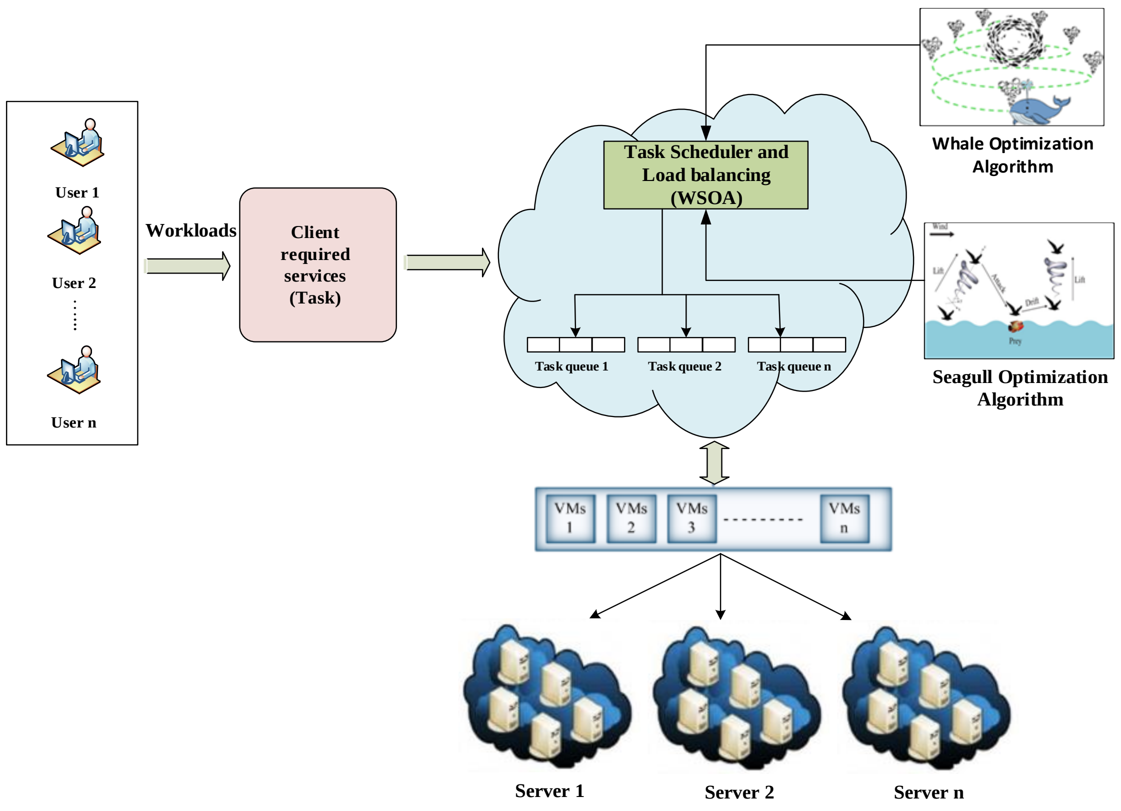
User 1
User 2
User n
Workloads
Client required services (Task)
Task Scheduler and Load balancing (WSOA)
Task queue 1
Task queue 2
Task queue n
Whale Optimization Algorithm
Seagull Optimization Algorithm
VMs 1
VMs 2
VMs 3
VMs n
Server 1
Server 2
Server n

Figure 1. Proposed architecture

2.1. System model

In cloud computing, there are several data centers, which are referred to as physical machines, and each data center has computational resources to perform tasks for users. On virtual machines, cloud users have different tasks to perform. Through the load balancing algorithm, various tasks are allocated to different VMs. Load balancing algorithms constantly monitor the load on VMs in the cloud infrastructure. The bandwidth, RAM, MIPS (million instructions per second), and the task's processing time determine the virtual machine's load. Because one operation's processing time differs from another's, the VMs load fluctuates. The goal of this technique is to plan tasks and assign tasks from overloaded VMs to underloaded VMs. As a result, task unbalanced situations throughout the overall system are avoided. This research aims to decrease the response time, processing time, and execution time value by queuing the most suitable set of tasks for virtual machines and improving CPU and memory utilization.

2.2. Mathematical model

Let F and M be the number of PMs in cloud C and the set of VMs on each PM.

$$C = \{PM_1, \dots, PM_K, \dots, PM_F\} \tag{1}$$

where C represents the cloud. Among the physical machines, PM_1 indicates the first, PM_K represents K^{th}, and PM_F indicates the F^{th}. PM_K is shown (2).

$$PM_K = \{V_1, \dots, V_i, \dots, Vm\} \tag{2}$$

Where the first virtual machine is V_1, the i^{th} VM is V_i, and the m^{th} virtual machine is V_m. In virtual machines, task processors, instructions for executing the user's tasks, memory, and task allocations indicate that task distributions among the under-loaded virtual machines are all included. The set of tasks on the virtual machines is T. In (3) shows the tasks processed by the VM in the cloud. If T number of tasks are processed by virtual computers in the cloud, the tasks are expressed as:

$$T = \{T_1, T_2, \dots, T_j, \dots, T_n\} \tag{3}$$

where the VM's total number of tasks is represented by T, the first and second tasks are denoted by T_1 *and* T_2, respectively. T_j denotes the j^{th} task, while T_n denotes the n^{th} task. The task T_j's execution time can be expressed as E_j, and the task T_j's priority level can be expressed as P_j.

2.3. Proposed load balancing approach

A cloud system comprises many data centers; each one comprises various PMs, and several virtual machines, as explained mathematically above. Multiple cloud users have submitted requests (tasks) to data centers. Following that, the VMs carry out those jobs. The proposed hybrid whale-seagull optimization algorithm (WSOA) assigns various requests to the VMs and numerous processes are included in the proposed LB system, such as; i) task scheduling, ii) calculation of VM capacity, iii) calculation of each load, and iv) hybrid WOA-SOA used for identifying the overloaded and under-loaded task and allocating overloaded VMs to under-loaded VMs.

2.3.1. Scheduling the task

The user will submit cloudlet list, a list of cloudlets that must be balanced to the scheduler:
Balancing: the appropriate VM is assigned based on the information collected.
VMCheck: check the VM state by comparing its capacity to its load using the three thresholds: upper, fair, and lower.
The capacity of VMs: a VM's capacity can be estimated based on its memory, MIPS, and processors. It is written as $C_{V\mathrm{m}}$, and it is shown mathematically in (4),

$$C_{VM} = Pe_{num} \times Pe_{mips} \tag{4}$$

where Pe_{num} is denoted as several computing units allocated in VM, Pe_{mips} is defined as the amount of MIPs.
Calculation of VM load: a VM's load is computed by combining two approximates: maximal and minimal bound approximates. Furthermore, the duration of execution of the task and the capacity of the VMs are taken into account. The load of a VM is calculated using two approximates: maximal bound and minimal bound, as well as the task's execution time and capacity. In (4) will be used to compute the VM load (VMload).

$$VM_{loadi} = \frac{\sum_{j=1}^{n} TL_j}{n} \tag{5}$$

Where TL refers to task length.

As a result of the previous process, the three thresholds were chosen. First threshold upper limited (TUL) is equal to 0.98. Secondly, threshold fair limited (TFL) is 0.7, and finally, threshold for lower limited (TLL) is 0.3, compared to VM load with a capacity to update VMs.

- *Stage 1.* If $\mathrm{VM}_{\mathrm{loadi}} <= C_{\mathrm{VMi}} \times \mathrm{TLL}$then VM state is labeled as underloaded.
- *Stage 2.* If$VM_{loadi} > C_{VMi} \times TLL$ then $\mathrm{VM}_{\mathrm{loadi}} <= C_{\mathrm{VMi}} \times \mathrm{TFL}$ then VM state labeled as a balanced state.
- *Stage 3.* If $VM_{loadi} > C_{VMi} \times TUL$ then VM state is labeled as overloaded.

After calculating the VM's capacity and load, hybrid WOA-SOA selects the best VM for task allocation.

2.3.2. Proposed SOA-WOA for task allocation

The proposed technique must locate suitable underloaded VMs for reallocating the task removed from the virtual machine, which is overloaded. The PMs may have multiple VMs that can complete the task. As a result, it is critical to look for the best VM for the job. Not only does the proposed technique perform LB, but it also considers the task's execution time, priority level, and response time. For determining the underloaded VMs, let us consider the VM's load and its capacity.

Foraging is an everyday activity for many animals in nature. Whales, for example, are known to engage in a unique form of foraging called bubble-net foraging. WOA designs downward spirals and shrinking encirclements based on whale predation behavior. When seagulls are looking for food, they migrate from one location to another. They spiral in the air to attack prey once they find rich prey. The hybrid WSOA optimizes global issues that combine the WOA's shrinking encircling mechanism with the SOA's spiral attack behavior, improving search capabilities locally and globally. Levy flight is a mechanism that controls local search by causing random walking behaviors. Seagull optimization, however, converges prematurely. Therefore, this paper incorporates the levy flight mechanism that encircles WOA's contraction and the SOA's local searching stage to increase the algorithm's ability to exploit and avoid early convergence. The following is the updated contraction near WOA's mechanism with levy flight strategy:

$$\overrightarrow{D_s} = levy(d) \times \left(levy(d) \times \overrightarrow{P_{bs}}(x) - \overrightarrow{P_s}(x)\right), \overrightarrow{P_s}(x+1) = \overrightarrow{P_{bs}}(x) - \vec{A} \times \overrightarrow{D_s} \quad (6)$$

where d is the current location vector's dimension, the current iteration number is denoted as x, and levy flight is defined as;

$$levy(x) = 0.01 \times \frac{r_1 \times \sigma}{|r_2|^{(1/\beta)}} \quad (7)$$

where r_1 and r_2 are random numbers in [0, 1], a constant value is β, and σ can be expressed as;

$$\sigma = \left(\frac{\Gamma(1+\beta) \times sin(\pi\beta/2}{\Gamma(1+\beta/2) \times \beta \times 2^{(\beta-1/2}}\right)^{(1/\beta)} \quad (8)$$

where $\Gamma(x) = (x-1)!$.
the SOA's spiral attack method has been improved in (9);

$$\overrightarrow{D_s} = \overrightarrow{C_s}(x) + B \times levy(d) \times \left(\overrightarrow{P_{bs}}(x) - \overrightarrow{P_s}(x)\right), \overrightarrow{P_s} = \left(\overrightarrow{D_s} \times x' \times y' \times z'\right) + \overrightarrow{P_{bs}}(x) \quad (9)$$

where reflection of the search agent's location that does not conflict with other search agents is denoted as C→s. →Ps denotes the search agent's current position, representing the distance the best-fit search agent and the search agent.

2.3.3. Fitness function

The fitness function of the proposed algorithm is given in the (10), and the algorithm of the proposed approach is given in Algorithm 1:

$$Fitness\ function = Min(Response\ time + Processing\ time) \quad (10)$$

Algorithm 1. Task allocation

```
Input: T = {$T_1$, $T_2$, $T_3$, $T_n$}
          VM = {$VM_1$, $VM_2$, $VM_3$, $VM_m$}
Output: The best potential task-to-VM mapping that is balanced
Step 1: Parameter initialization
        Initialize the number of tasks, number of VMs, parameters, and the maximum number of
        iterations $Max_{itr}$.
Step 2: Evaluate the workload of the best hyper-heuristic solution by determining the load
and capacity on each VM by,
                    Load on VM($LVM_{i,t}$) = $\frac{N(T,t)}{S(VM_{i,t})}$
                    Where, $LVM_{i,t}$ = Load of $VM_i$ at time t
                    $N(T,t) = set\ of$ task $at\ time\ t\ on\ service\ queue$
                    $S(VM_{i,t}) = Rate\ of\ service\ rate\ of\ VM_i\ at\ time\ t$
Step 3: To keep track of the presence of low-load VMs, such as to test the load on the VM,
        followed by WOA to optimize the SOA for assigning a task.
Step 4: Determine which VMs to be allocated based on the load.
Step 5: If nodes are not overloaded, look for underloaded nodes. Upon finding an underloaded
user, a virtual machine is allocated, and the process continues until no underloaded VMs are
found.
Step 6: Find VM to transfer the task
Step 7: Task Migration based on VM group
Step 8: Continue until the whole task queue is completed.
As a result of its load scheduling capabilities, the WSOA reduces response times for cloud
requests. By utilizing a more comprehensive search area, users get satisfaction. The results
are presented in the upcoming sections.
```

3. RESULTS AND DISCUSSION

The experimental setup and evaluation of the proposed algorithm are presented in this section, along with a comparison to some other recent algorithms. To determine the effectiveness of the proposed algorithm, the CloudSim simulator is used with Intel® Core™ CPU, 2.60 GHz, 8G RAM. Cloud platform components include cloud clients, cloudlets (undertakings), client, cloudlets, datacenter and VM. These parameters are expressed as a constant value for stable stimulation and as a lower limit for the simulation model, with the possibility of increasing those values during execution. The data center allocates CPUs, memory, and storage to each virtual machine. Table 1 shows the main components and parameters of our simulation.

Table 1. Parameter setting for cloud simulator

Entities	Parameters	Values
Datacenter	Number of CPUs	25
	Storage	10 TB
	Number of data centers	1
	Processing capacity	10,000 MiPs
	RAM capacity	50 GB
Task properties	Number of tasks	100-1,000
	Size of the tasks	30,000-300,000
	CPU length	[100, 1,000] MiPS
Virtual machine	Number of CPUs	1
	Bandwidth	200 MIPS
	Capacity of RAM	1,028 MB
	Capacity of storage	100 GB

3.1. Performance metrics

Several performance metrics are utilized to analyze the effectiveness of the proposed task scheduling technique, such as response time, throughput, execution time, resource utilization, and reliability. The mathematical formulation of these metrics is defined in the following;

1. Response time: average response time:

$$T_{resp} = \frac{\sum_{j=1}^{n} r_j}{n} \tag{11}$$

2. Throughput: throughput time is calculated using in (12)

$$Throughput\ time = \frac{m}{max_{1\leq i<m}\{FT_i\}} \tag{12}$$

3. Execution time: the execution time of the proposed approach is computed using in the (13).

$$Execution\ time = \frac{\text{Task}}{\text{Processing speed of VM}} \tag{13}$$

4. Resource utilization: the server resource utilization displays how much time the resources in the cloud data center should be rescheduled to complete the tasks.

$$util_{Re\,s}(t) = Dem_{Re\,s} \times \lambda(t) \tag{14}$$

5. Reliability: the quality-of-service (QoS) reliability domain R is defined as

$$T_R = Loss \times Period \times Burstiness \times Corruption, \text{ with Loss} = n_0, \text{ Period} = r+, \text{ Burstiness} = r+ \text{ and corruption} = \{r \in r | 0 \leq r \leq 100 \tag{15}$$

3.2. Comparative analysis

The proposed approach is compared to other heuristic algorithms in this section with HHO, hybrid dragonfly and firefly algorithm (ADA), spider monkey algorithm (SMA), and bird swarm optimization (BSO-LB) since they have similar goals. These heuristic algorithms aim to improve cloud task scheduling and balance the load. The proposed method's performance is assessed using a variety of QoS metrics such as CPU utilization, execution time, reliability, processing time, response time, throughput, and memory utilization. The results of the evaluation are as follows. The task sizes are produced randomly at runtime in Table 2 of the experiment, and the size is expressed as millions of instructions per second (MIPs).

Table 2. Size of synthetic datasets

Tasks types	Number of tasks
Small	100-200
Medium	400-500
Large	600-700
Extra-large	800-1,000

3.3.1. Response time

Listed are the times it takes to complete various tasks. Compared to other existing algorithms, fewer requests are queued, and device response time has decreased with the hybrid WSOA algorithm, as shown in Figure 2. Thus, the proposed algorithm is more suitable for balancing the load than existing algorithms. Compared to the BSO-LB algorithm, the response time of ADA and SMA is higher. Because the SMA algorithm is highly reliant on the hyperparameters and has minimal convergence. Moreover, the ADA becomes trapped in local optima because of the algorithm's high rate of exploitation.

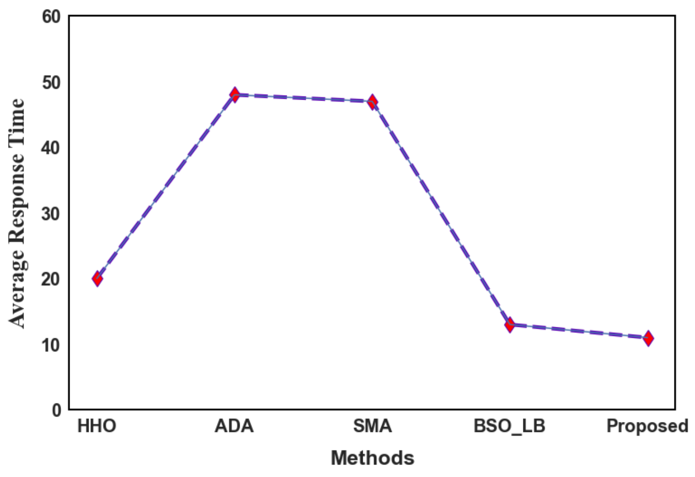

Figure 2. Comparative analysis of response time

3.3.2. Execution time

Figure 3 depicts the execution time (in seconds) calculated for various jobs. The task execution time is determined by the VM instances chosen. When tasks are assigned to the most powerful VM instances with the highest resource capacity, the task execution time is reduced compared to less powerful VM instances. HHO and ADA techniques require a maximum average execution time of 103 and 96 milliseconds, respectively. SMA and BSO-LB methods required a slightly lower average execution time, with an average of 87 ms and 84 ms for 1,000 tasks, respectively. However, for a maximum number of tasks, the proposed WSOA algorithm produced efficient outcomes with the shortest average execution time of 81 ms.

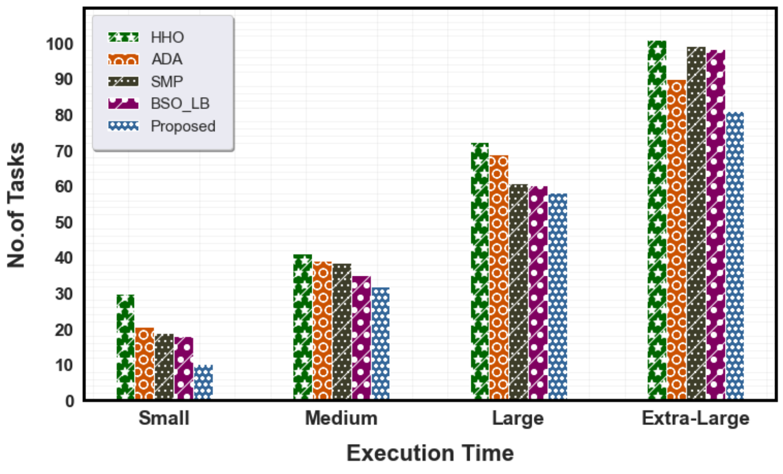

Figure 3. No. of tasks vs execution time

3.3.3. Processing time

Figure 4 depicts the experimental outcomes of the approaches. When comparing the outcomes in terms of processing time, it is worth noting that the HHO and ADA approaches took maximum average processing times of 12,500 ms and 11,500 ms, respectively. SMA and BSO-LB techniques required a slightly shorter average processing time, with average response times of 11,250 ms and 11,000 ms, respectively. However, for 1,000 tasks, the proposed WSOA algorithm produced an effective result with the shortest average reaction time of 10,700 ms.

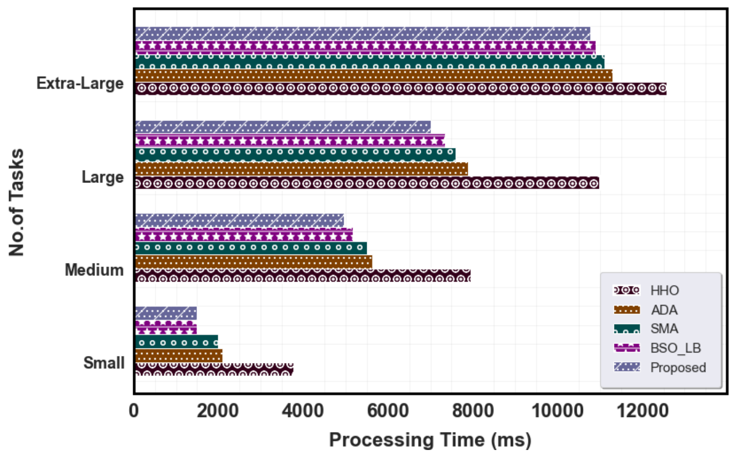

Figure 4. No. of tasks vs processing time

3.3.4. Resource utilization

Based on CPU and memory consumption, we analyzed resource utilization. Figure 5 depicts the results of the CPU utilization evaluation for all approaches. The WSOA reached the best CPU utilization across all tasks compared to other techniques. HHO, ADA, SMA, and BSO-LB models got lower CPU utilization by obtaining the lowest utilization under small types of tasks of about 52%, 57%, 67%, and 69%, respectively. However, the provided model outperformed the competition by achieving maximum CPU usage of 98% for extra-large jobs and 72% for minor tasks.

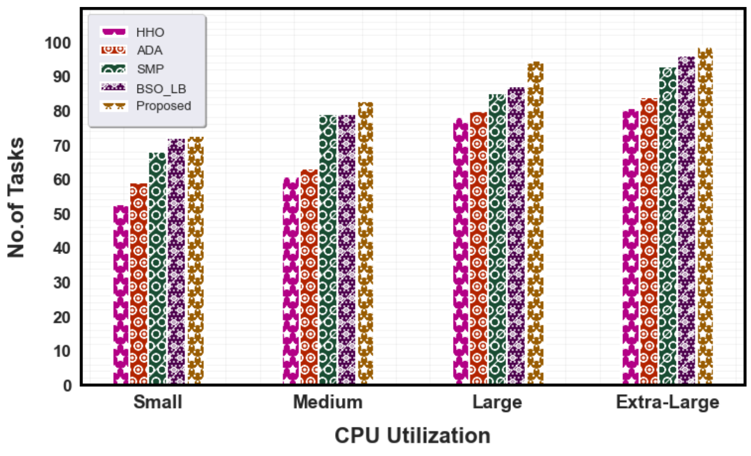

Figure 5. No. of tasks vs CPU utilization

Figure 6 depicts a detailed comparison WSOA and existing approaches in terms of memory usage. Compared to other approaches, the WSOA obtained the highest memory usage in various jobs see Figure 6. HHO, ADA, SMA, and BSO-LB models have low CPU consumption under small tasks, with the lowest usage of 47%, 52%, 57%, and 60%, respectively. However, the provided model outperformed the competition by achieving the highest memory usage of 93% for extra-large jobs and % for minor tasks, respectively.

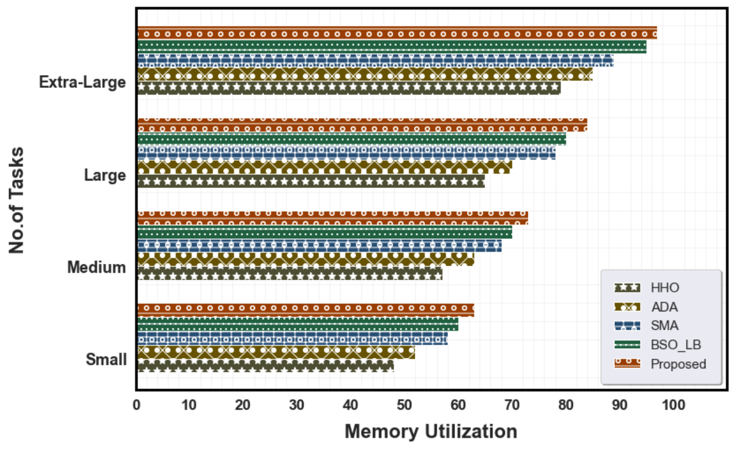

Figure 6. No. of tasks vs memory utilization

3.3.5. Reliability

The proposed method's reliability is compared to the other alternatives depicted in Figure 7. HHO, ADA, SMA, and BSO-LB models have poor dependability in minor activities, with minimum reliability of 47%, 56%, 66%, and, 69% respectively. The proposed WSOA models, on the other hand, outperformed the competition by achieving maximum reliability of 82% for extra-large workloads. This is because the resources are assigned to jobs while considering their failure rate. Moreover, tasks are running on the processor's usual frequency, which has the highest level of reliability.

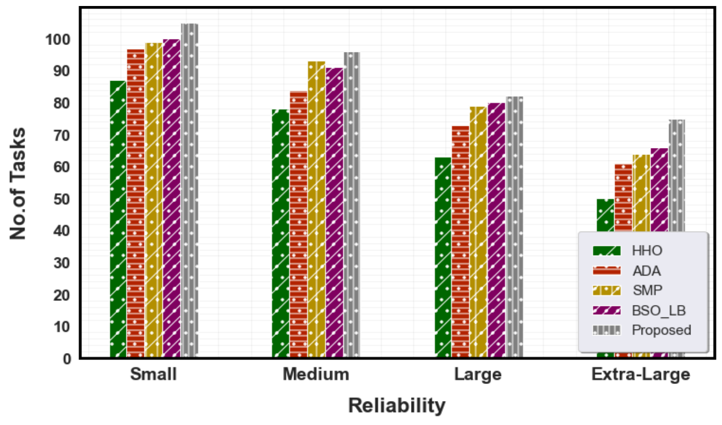

Figure 7. No. of tasks vs reliability

3.3.6. Throughput

Figure 8 depicts the experimental outcomes of the procedures. HHO and ADA models had low average throughput in small tasks, with a minimum average throughput of 82% and 91%, respectively. Simultaneously, the SMA and BSO-LB approaches attempted to keep up by slightly improving average throughput of 97% and 99%, respectively. However, the proposed model outperformed the competition by achieving an average throughput of 82% for extra-large tasks.

In addition, the proposed strategy has been compared using various numbers of virtual machines. For this comparison, the algorithm has been run ten times, and the average results are shown in Figure 9. From the figure, it is observed that the proposed algorithm generates a high load balancing level compared to other existing techniques. Especially in 450 and 800 VMs, it effectively balances the workload and attains superior results than other compared techniques. Compared to all other techniques, the performance of HHO is substandard. Due to its population diversity problem, it cannot effectively handle the load balancing problem.

Figure 10 displays the comparative experiment to show the efficiency of the proposed algorithm on comparing with other existing approaches. It shows that the efficiency of HHO is about 87%, the efficiency of ADA is about 91%, SMA is about 93%, BSO-LB is about 96%, and the Proposed algorithm outcomes the other algorithm with an efficiency of about 98.5%. This result is achieved by including a levy flight strategy into the proposed approach that encircles WOA’s contraction and the SOA’s local searching stage to increase the algorithm's ability to exploit and avoid early convergence.

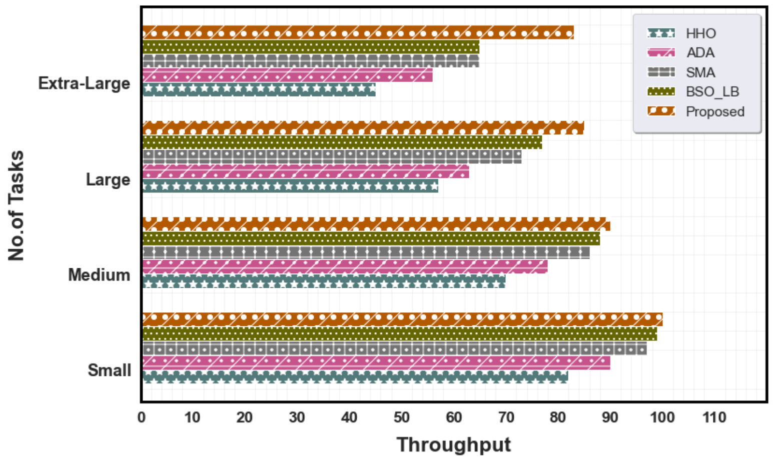

Figure 8. No. of tasks vs throughput

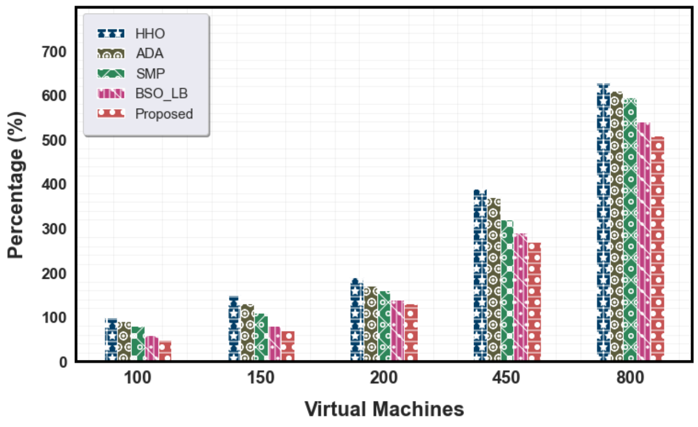

Figure 9. Comparative analysis for various VMs

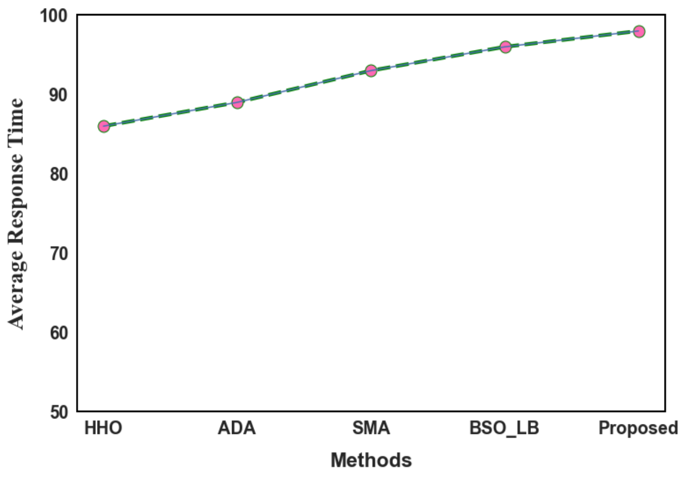

Figure 10. Performance comparison in terms of efficiency

As the data reveal, when compared to other methods, the proposed method consistently produces the best outcomes. It is owing to the benefits of both WOA and SOA algorithms functioning together. Many existing algorithms have drawbacks that make them unsuitable for specific situations, and load balancing is ineffective, such as spider monkey optimization (SMO) and bird swarm algorithms are affected by premature convergence. Moreover, firefly's limited exploratory capacity makes it occasionally hard to find the best answers. Low solution accuracy, easy stalling at local optima, and an imbalance exploration and exploitation may be the effects of the DA algorithm's extensive social interactions. Because of these drawbacks, load balancing and work scheduling using present methodologies are not done correctly.

In contrast to prior methods, the proposed model developed an improved optimization-based scheduling approach to address the issues of system overhead, stability, reliability, and poor allocation of resources. The hybrid WOA-SOA algorithm is used in the proposed model to balance the load and schedule the work. The WSOA benefits from the powerful global search capability of WOA and the quick convergence capability of SOA. Also included in the SOA search formula is the levy fly strategy, which can better balance exploitation and exploration techniques and prevent early convergence of techniques. This benefit motivates the suggested method to address load balancing issues by correctly allocating resources across VMs based on the best fitness function. Furthermore, it successfully addresses the issue of inefficient resource allocation and increases system overhead. With effective load balancing and meeting the goals of both cloud users and providers, it can be said that the proposed solution performed better than the alternatives in all the factors considered.

4. CONCLUSION

Infrastructure-based service providers face a constant challenge in task scheduling and load balancing since they serve more consumers while using fewer VMs. More research techniques for scheduling and load balancing were offered, but they had limitations, such as not considering the runtime variation of task parameters and VM properties. To address this problem, this research proposed a metaheuristic optimization technique that combined the scheduling of tasks with load balancing. It is achieved by combining the WOA and SOA. The WSOA algorithm given here established an adequate average load for making and improving critical measures such as optimal resource consumption and job response time. For simulation, the number of measures used to demonstrate the results was the utilization of resources, response time, execution time, reliability, throughput, and processing time, and the proposed method achieves 12 ms response time, 81 ms execution time, 98% resource utilization, 69% reliability, and 82% throughput. The proposed WSOA model performed better than the other methods based on performance, according to the results of the experiments. The proposed method will be improved in the future by incorporating other meta-heuristics algorithms, and QoS characteristics like energy consumption, migration time, and optimization time will also be considered. Additionally, future research will also take into account overlapping task collections and multi-objective approaches.

REFERENCES

[1] G. Sreenivasulu and I. Paramasivam, "Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing," *Evolutionary Intelligence*, vol. 14, no. 2, pp. 1015–1022, Jun. 2021, doi: 10.1007/s12065-020-00517-2.

[2] M. Yadav and S. Gupta, "Hybrid meta-heuristic VM load balancing optimization approach," *Journal of Information and Optimization Sciences*, vol. 41, no. 2, pp. 577–586, Feb. 2020, doi: 10.1080/02522667.2020.1733190.

[3] K. Kamakshaiah, K. V. Rao, and M. Subrahmanyam, "SABE: efficient and scalable-filtered access control in distributed cloud data storage," in *Smart Innovation, Systems and Technologies*, vol. 78, 2018, pp. 31–42.

[4] M. Junaid, A. Sohail, A. Ahmed, A. Baz, I. A. Khan, and H. Alhakami, "A hybrid model for load balancing in cloud using file type formatting," *IEEE Access*, vol. 8, pp. 118135–118155, 2020, doi: 10.1109/ACCESS.2020.3003825.

[5] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 6, pp. 2332–2342, Jun. 2022, doi: 10.1016/j.jksuci.2020.01.012.

[6] B. Muthulakshmi and K. Somasundaram, "A hybrid ABC-SA based optimized scheduling and resource allocation for cloud environment," *Cluster Computing*, vol. 22, no. S5, pp. 10769–10777, Sep. 2019, doi: 10.1007/s10586-017-1174-z.

[7] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Computing*, vol. 26, no. 5, pp. 2479–2488, Oct. 2021, doi: 10.1007/s10586-020-03221-z.

[8] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: performance evaluation," *Computer Networks*, vol. 162, p. 106860, Oct. 2019, doi: 10.1016/j.comnet.2019.106860.

[9] X. Wei, "Task scheduling optimization strategy using improved ant colony optimization algorithm in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, Oct. 2020, doi: 10.1007/s12652-020-02614-7.

[10] S. M. G. Kashikolaei, A. A. R. Hosseinabadi, B. Saemi, M. B. Shareh, A. K. Sangaiah, and G. B. Bian, "An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm," *Journal of Supercomputing*, vol. 76, no. 8, pp. 6302–6329, Aug. 2020, doi: 10.1007/s11227-019-02816-7.

[11] P. Zhang, M. C. Zhou, and X. Wang, "An intelligent optimization method for optimal virtual machine allocation in cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1725–1735, Oct. 2020, doi: 10.1109/TASE.2020.2975225.

[12] P. Xu, G. He, Z. Li, and Z. Zhang, "An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization," *International Journal of Distributed Sensor Networks*, vol. 14, no. 12, p. 155014771879379, Dec. 2018, doi: 10.1177/1550147718793799.
[13] G. Natesan and A. Chokkalingam, "Task scheduling in heterogeneous cloud environment using mean grey wolf optimization algorithm," *ICT Express*, vol. 5, no. 2, pp. 110–114, Jun. 2019, doi: 10.1016/j.icte.2018.07.002.
[14] P. Krishnadoss and P. Jacob, "OLOA: based task scheduling in heterogeneous clouds," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 1, pp. 114–122, Feb. 2019, doi: 10.22266/ijies2019.0228.12.
[15] G. Li and Z. Wu, "Ant colony optimization task scheduling algorithm for SWIM based on load balancing," *Future Internet*, vol. 11, no. 4, p. 90, Apr. 2019, doi: 10.3390/fi11040090.
[16] N. Dordaie and N. J. Navimipour, "A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments," *ICT Express*, vol. 4, no. 4, pp. 199–202, Dec. 2018, doi: 10.1016/j.icte.2017.08.001.
[17] L. Abualigah and A. Diabat, "A novel hybrid antlion optimization algorithm for multi-objective task scheduling problems in cloud computing environments," *Cluster Computing*, vol. 24, no. 1, pp. 205–223, Mar. 2021, doi: 10.1007/s10586-020-03075-5.
[18] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, "IPSO task scheduling algorithm for large scale data in cloud computing environment," *IEEE Access*, vol. 7, pp. 5412–5420, 2019, doi: 10.1109/ACCESS.2018.2890067.
[19] S. Torabi and F. Safi-Esfahani, "A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing," *Journal of Supercomputing*, vol. 74, no. 6, pp. 2581–2626, Jun. 2018, doi: 10.1007/s11227-018-2291-z.
[20] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid heuristic-metaheuristic techniques in cloud environment," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 813–824, Mar. 2022, doi: 10.1016/j.jksuci.2019.02.010.
[21] G. Muthsamy and S. R. Chandran, "Task scheduling using artificial bee foraging optimization for load balancing in cloud data centers," *Computer Applications in Engineering Education*, vol. 28, no. 4, pp. 769–778, Jul. 2020, doi: 10.1002/cae.22236.
[22] S. Pang, W. Li, H. He, Z. Shan, and X. Wang, "An EDA-GA hybrid algorithm for multi-objective task scheduling in cloud computing," *IEEE Access*, vol. 7, pp. 146379–146389, 2019, doi: 10.1109/ACCESS.2019.2946216.
[23] G. A. P. Princess and A. S. Radhamani, "A hybrid meta-heuristic for optimal load balancing in cloud computing," *Journal of Grid Computing*, vol. 19, no. 2, p. 21, Jun. 2021, doi: 10.1007/s10723-021-09560-4.
[24] S. Rani and P. K. Suri, "An efficient and scalable hybrid task scheduling approach for cloud environment," *International Journal of Information Technology (Singapore)*, vol. 12, no. 4, pp. 1451–1457, Dec. 2020, doi: 10.1007/s41870-018-0175-3.
[25] M. R. Thanka, P. U. Maheswari, and E. B. Edwin, "A hybrid algorithm for efficient task scheduling in cloud computing environment," *International Journal of Reasoning-based Intelligent Systems*, vol. 11, no. 2, pp. 134–140, 2019, doi: 10.1504/IJRIS.2019.099850.
[26] A. Pradhan and S. K. Bisoy, "A novel load balancing technique for cloud computing platform based on PSO," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 7, pp. 3988–3995, 2022, doi: 10.1016/j.jksuci.2020.10.016.

BIOGRAPHIES OF AUTHORS

Ravinder Bathini is an assistant professor in Department of Information Technology, Vardhaman College of Engineering, Kacharam, Shamshabad-501218 Hyderabad, Telangana, India and research scholar in K L University, Green Fields, Vaddeswaram, Andhra Pradesh 522502. His research area includes green cloud computing, cloud analytics, cloud deployment models and cloud security. He can be contacted at email: bravinder552@gmail.com.

Naresh Vurukonda is associate professor Department of Computer Science and Engineering at KL University, Green Fields, Vaddeswaram, Andhra Pradesh 522502. His main research subject is cloud computing and big data. He can be contacted at email: nareshvurukonda789@gmail.com.