Hindawi
Scientific Programming
Volume 2023, Article ID 8830895, 11 pages
https://doi.org/10.1155/2023/8830895

*Research Article*

# SLA Aware Task-Scheduling Algorithm in Cloud Computing Using Whale Optimization Algorithm

**Sudheer Mangalampalli**,[1] **Sangram Keshari Swain**,[2] **Ganesh Reddy Karri**,[1] **and Satyasis Mishra**[3]

[1]*School of Computer Science and Engineering, VIT-AP University, Amaravati, AP, India*
[2]*Centurion University of Technology and Management, Sitapur, Odisha, India*
[3]*Adama Science and Technology University, Adama, Ethiopia*

Correspondence should be addressed to Satyasis Mishra; satyasismishra@gmail.com

Received 20 January 2023; Revised 22 March 2023; Accepted 12 April 2023; Published 20 April 2023

Academic Editor: Jiangbo Qian



Task scheduling in Cloud Computing paradigm poses new challenges for cloud provider as heterogeneous, diversified tasks arrived on to cloud console. To schedule these type of tasks efficiently on to virtual resources in cloud paradigm, an effective scheduler is needed, which precisely maps tasks to virtual machines by considering priorities of both tasks and VMs. Existing scheduling algorithms failed to map tasks precisely to virtual resources due to high dynamic nature in cloud environment which leads to increase of makespan and SLA violations will be increased. In this paper, authors proposed a task-scheduling mechanism, which considers task priorities and VMs. To model this scheduling paradigm we have chosen whale optimization through which our scheduler will take decisions for scheduling tasks precisely onto virtual resources in cloud environment. Entire simulation was carried out on CloudSim. Initially we have chosen random generated workload to run simulation and after that, we have considered a real-time workload named as BigDataBench and ran our simulation. Finally, we compared our proposed work with classical baseline mechanisms. From simulations we observed that proposed whale scheduler improved makespan for PSO, ACO, GA, and W-schedulers by 20.07%, 17.55%, 19.9%, and 6.35%, respectively, and 17.3%, 17.86%, 17.64%, and 5.93%, respectively, for BigDataBench workloads. SLA violations improved over PSO, ACO, GA, and W-Scheduler by 56.76%, 42.17%, 35.29%, and 24.53%, respectively, and 63.42%, 23.33%, 55.51%, and 40.1%, respectively, for BigDataBench workloads. From extensive simulation results, our proposed scheduler using whale optimization approach minimizes makespan and SLA violations to a great extent.

## 1. Introduction

Cloud Computing model gave a new hope to entire IT industry, as well as to other industries such as education, healthcare, and government sectors to accommodate their computing and storage infrastructure with different cloud services. Existing on premises infrastructure cannot accommodate huge computations and storage requirements of various industries as data are evolving in a huge manner from various resources, and therefore, to handle various storage and computing requirements of various users concurrently with existing infrastructure is a vain. Therefore, to handle these many users and their requests concurrently many of the companies migrating towards cloud environment, which gives on demand access to a network, which consists of a shared pool of configurable resources on paid bases, based on user requests [1]. To handle these large numbers of users around globe cloud paradigm uses different types of deployment models, i.e., public cloud leverages all different services to all users around the world publicly on paid basis. Private cloud-leverages services to all cloud services belong to a specific organization. Hybrid cloud-leverages services, which are restricted to some of the users in organization, and some of the services extended to

all users around the world [2]. All facilities in cloud paradigm provides access to users by providing virtual resources through various service models [3] for providing computing infrastructure, storage, and network as services to users based on user requirement. To provision and deprovision virtual resources based on requirement of users Cloud Computing environment need an effective scheduler, which should map incoming heterogeneous and diversified tasks onto appropriate virtual resources by minimizing makespan and SLA violations which are integral part of any cloud paradigm. Many of existing task-scheduling algorithms proposed by authors addressed various parameters, makespan, energy consumption, and execution cost. Existing authors apply various nature-inspired algorithms, i.e., PSO [4] and ACO [5] but both of these approaches have their own limitations as PSO faces a problem and it cannot explore solution space and trapped into a local optimum, whereas ACO cannot handle dynamic population in solution space but cloud computing is a paradigm which consists of dynamic, diversified and heterogeneous requests from various users. Therefore, an efficient task scheduler need to be formulated which maps tasks effectively by considering type of requests, thereby calculating their priorities and finally need to map these requests to suitable VMs while minimizing makespan and SLA violations. In this manuscript, we used whale optimization to model our scheduler, which addresses parameters, makespan and SLA violation. The main motivation to conduct this research is to minimize number of SLA violations in Cloud Computing paradigm while provisioning or scheduling tasks to appropriate virtual resources and if SLA made between Cloud provider and Cloud user is violated in terms of services then there will be loss on both the ends. Therefore, we considered this as primary motivation to conduct this research.

Highlights and main contributions of our manuscript are presented as follows:

(i) Effective task-scheduling mechanism proposed using consideration of priorities of tasks and VMs

(ii) Whale optimization is used to model task-scheduling mechanism

(iii) Extensive simulations conducted on CloudSim [6], workloads considered in algorithm are randomly generated workload and real time BigDataBench workloads

(iv) Makespan and SLA violation are addressed as parameters

Below section discusses briefly about various related works conducted by researchers using metaheuristic and nature-inspired algorithms.

## 2. Related Works

The authors formulated task-scheduling mechanism in [7] aims at parameters, i.e., makespan, throughput, and resource utilization. Adaptive PSO used as methodology for scheduling problem, which dynamically balance incoming workloads, based on inertia weights. Extensive experiments were conducted by using CloudSim. Workload was taken from [8], a benchmark dataset to evaluate efficacy of approach. It evaluated against different variants of PSO, and from results, it proved that it outperforms all compared approaches for improvement in makespan, throughput, and utilization of resources by 10%, 12%, and 60%, respectively. In [9] authors formulated a task-scheduling mechanism for balancing load in cloud. MCFCMA and PSO algorithms are used to model scheduling algorithm. Initially tasks are selected based on their load and cluster those tasks with MCFCMA approach and scheduling of tasks to VMs based on PSO approach. Entire simulation is implemented on CloudSim. It is compared aganist metaheuristic algorithms. From results, it proved that load balance of tasks and scheduling improved over existing variations of PSO. In [10], a deadline aware task-scheduling approach using multiobjective function were developed for heterogeneous workloads. Adaptive PSO-RADL algorithm was used as methodology based on dynamic inertia weights assignment in algorithm. It was implemented on CloudSim, and generation of workload was carried out with synthetic and real-time benchmark datasets. It was evaluated against different metaheuristic approaches. From results, PSO-RDAL showed huge impact over existing approaches for makespan, response time, resource utilization, penalty cost, and total execution cost. In [11], multiobjective scheduling mechanism developed which aims at parameters, i.e., makespan, execution time, execution cost, and energy consumption. A hybrid approach, i.e., CR-PSO was used as methodology for scheduling. Simulation was carried out on CloudSim. It was evaluated over different metaheuristic approaches. From results, it proved that a significant reduction was observed for specified parameters. In [5], a multiobjective scheduling mechanism was developed using combination of GA and ACO approaches. GA and ACO algorithms hybridized to model task-scheduling approach. Simulations were carried out on CloudSim. It was compared over GA and ACO algorithms, and from results, it observed that throughput, task completion time, and response time significantly minimized with GA-ACO approach. In [12], a hybrid task-scheduling algorithm was developed to address parameters, makespan, utilization of resources, and total computation cost. It was modeled by using a hybrid approach PSO-ACO. PSO was used for generating decisions at a global level, and ACO was used for generating decisions at a local level. Experiments were conducted on CloudSim. It was evaluated against variants of PSO and ACO algorithms. Experimental data revealed that hybrid approach showed impact over variants. In [13], hybrid task scheduling was formulated to address parameters, makespan and overall cost. A hybrid approach with nature-inspired algorithms was used for modeling task scheduler. Cuckoo and crow search algorithms were used to model scheduling algorithm. It was simulated on CloudSim tool. It was compared over MO-ACO, Min-Min, and ACO algorithms, and the results showed that CCSA outperforms existing algorithms. In [14], scheduling approach designed by updating pheromone leads to increase in acceleration of ant exploration in solution space. MOTS-ACO was used as methodology for designing scheduling problem. It was

implemented on CloudSim. It was compared over existing scheduling algorithms modeled various metaheuristic approaches, and finally, simulation results revealed that MOTS-ACO outperforms existing approaches for makespan, turnaround time, and consumption of power. In [15], a hybrid task scheduling was developed to focus on task completion time and load balance of tasks. EDA-GA was used for scheduling mechanism. It was simulated on CloudSim and was compared against existing EDA and GA approaches. From results, hybrid approach proved that it outperforms existing EDA and GA algorithms for specified parameters. In [16], an energy efficient scheduling mechanism was formulated to minimize energy consumption. Aim of this scheduling approach is for identifying appropriate virtualized execution environment for a task while minimizing energy consumption. It was implemented on a customized simulation environment. It was compared against existing RC-GA, AMTS, and E-PAGA approaches, and from results, EPETS outperforms existing mechanisms for a specified parameter. In [17], a scheduling mechanism was devised to determine effective task transfer time. This task transfer time calculated based on task capacity, size of task, number of tasks, number of VMs, and throughput. MVO-GA was used as methodology for scheduling in this scenario. Simulations were carried out on MATLAB, and effectiveness of task transfer time with respect to all the aforementioned parameters were identified, and MVO-GA showed its effectiveness for task transfer time. In [18], a scheduling algorithm was formulated for customer satisfaction where quality of service and makespan were addressed as parameters in multi-cloud environment. GA-based customer satisfaction framework was developed, where in first phase resource allocation and task scheduling were based on shortest jobs to generate scheduling decisions. All the simulations were carried out on MATLAB, and from results, GACCRATS proved that makespan and customer satisfaction can improve in multi-cloud environment. In [19], authors formulated a multi-objective scheduling algorithm addresses parameters, makespan, response time, and QoS. Genetic algorithm modified and added greedy search for methodology of this approach. CloudSim was used for simulation and evaluation over greedy search, GA. The results revealed that MGGS show impact over baseline approaches. In [20], a hybridized approach was formulated using OBL and CS algorithms to address parameters, makespan and cost. OBL was used as global search and CS was used as local search for this approach. It was evaluated over PSO, IDEA, and GA approaches. OCSA outperforms existing mechanisms for specified parameters.

From Table 1, we can clearly observe that many of task-scheduling algorithms formulated using various metaheuristic approaches and they suffer to provide accurate solutions by assigning appropriate virtual resources to incoming requests by various users. Many of existing authors addressed parameters, i.e., makespan, execution cost, execution time, and energy consumption, but many authors ignored SLA violations as a parameter, even some authors addressed SLA violation as a metric they failed to achieve mapping tasks to suitable VMs for consideration priorities of tasks and VMs. Therefore, we considered whale optimization algorithm as methodology for scheduling in cloud paradigm while minimizing SLA violations and makespan.

## 3. Problem Definition and System Architecture

This section discusses problem definition and proposed system architecture used for scheduling. Consider that we assumed set of $k$ tasks indicated as $t_k = \{t_1, t_2, t_3, \ldots\ldots t_k\}$.set of $n$ VMs indicated as $vm_n = \{vm_1, vm_2, \ldots\ldots.., vm_n\}$, set of $i$ hosts indicated as $H_i = \{H_1, H_2, H_3, \ldots.., H_i\}$, and set of $j$ datacenters as $D_j = \{D_1, D_2, \ldots\ldots, D_j\}$. Therefore, problem is defined in a way that $k$ tasks are mapped on to *nvms* which are resided in $H_i$ hosts in turn resided back with $D_j$ datacenters by considering priorities of tasks and VMs, while minimization of makespan, and SLA violations. Table 2 shows notations used in proposed system architecture.

Figure 1 indicates proposed system architecture. Initially, concurrent user requests put forward to cloud administrative console and broker on behalf of cloud users submits them to task manager. Task manager verify validity of user requests coming onto cloud interface by concerning with SLA. If they are valid requests, it will pass those requests to waiting queue before passing it to task scheduler. In this architecture, after validating requests from task manager it calculates priorities of all diversified and heterogeneous tasks based on size and run time capacity of tasks. After calculation of task priorities, VM priorities are calculated based on unit cost electricity price. Based on the calculation of priorities from tasks and VMs, they will be fed to the waiting queue and it will map the highest prioritized task to a highest prioritized VM. While mapping these requests according to these priorities scheduler minimize makespan and SLA violations.

Initially to evaluate priorities of tasks, we calculated workload on all VMs. Workload on all VMs was indicated by using the following equation:

$$lo_{vm_n} = \sum lo^n. \tag{1}$$

$lo^n$ represents workload on $n$ VMs.

These VMs resided in set of $H_i$ hosts. Therefore, entire workload on hosts is calculated by using the following equation:

$$lo_{H_i} = \frac{lo_{vm_n}}{\sum H_i}, \tag{2}$$

where $lo_{H_i}$ represents workload on $H_i$ hosts.

We need to verify whether user requests or tasks can processed on to a certain VM. For this purpose, processing capacity of a VM has to be defined and it is indicated by using the following equation:

$$\text{pro}_{ca_{vm}} = \text{pro}_{\text{no}} * \text{pro}_{\text{MIPS}}, \tag{3}$$

where $\text{pro}_{\text{no}}$ represents processing elements and $\text{pro}_{\text{MIPS}}$ represents processing capacity based on number of instructions processed per second.

Table 1: Summary of task-scheduling algorithms using various metaheuristic approaches.

| References | Methodology | Objectives of resource scheduling algorithms modeled by different nature-inspired algorithms |
|---|---|---|
| [7] | Adaptive PSO | Makespan, throughput, resource utilization |
| [9] | MCFCM-PSO | Load balancing, makespan |
| [10] | PSO-RDAL | Makespan, response time, penalty cost, total execution cost |
| [11] | CR-PSO | Makespan, execution time, execution cost, energy consumption |
| [5] | GA-ACO | Response time, task completion time, throughput |
| [12] | PSO-ACO | Makespan, resource utilization, total computation cost |
| [13] | CCSA | Makespan, overall cost |
| [14] | MOTS-ACO | Makespan, turnaround time, power consumption |
| [15] | EDA-GA | Task completion time, load balance of tasks |
| [16] | EPETS | Energy consumption |
| [17] | MVO-GA | Task transfer time |
| [18] | GACCRATS | Makespan, customer satisfaction |
| [19] | MGGS | Makespan, response time, QoS |
| [20] | OCSA | Makespan, cost |
| [24] | CGA | Task completion time, total execution cost |
| [25] | IWC | Task scheduling time, scheduling cost |
| [26] | SLNO | Energy, power consumption, resource utilization |
| [27] | GCWOAS2 | Task completion time, load balance of virtual resources |
| [28] | GAGELS | Makespan, resource utilization |
| [29] | DILS | Makespan, learning rate |

Table 2: Notations used in system architecture.

| Notation | Meaning |
|---|---|
| $t_k$ | Set of tasks |
| $\mathrm{vm}_n$ | Set of virtual resources |
| $H_i$ | Set of hosts |
| $D_j$ | Set of datacenters |
| $lo_{vm_n}$ | Workload on virtual resources |
| $lo_{H_i}$ | Workload on physical host |
| $\mathrm{pro}_{\mathrm{ca}_{\mathrm{vm}}}$ | Processing capacity of a virtual resource |
| $t_{\mathrm{pr}_k}$ | Priorities of $k$ tasks |
| $\mathrm{vm}_{\mathrm{pr}_n}$ | Priorities of $n$ virtual resources based on unit cost electricity |
| $\mathrm{ms}^k$ | Makespan for set of $k$ tasks |
| $\mathrm{SLA}_{\mathrm{vio}}$ | SLA violation |

For mapping of tasks to precise, VMs scheduler need to know the size of the task and it is calculated using the following rquation:

$$t_k^{\mathrm{len}} = t^{\mathrm{MIPS}} * t_k^{\mathrm{pr}}. \tag{4}$$

Now priorities of all tasks are calculated using the following equation:

$$t_{pr_k} = \frac{t_k^{\mathrm{len}}}{\mathrm{pro}_{ca_{vm}}}. \tag{5}$$

VM priorities using unit electricity cost are calculated using the following equation:

$$\mathrm{vm}_{\mathrm{pr}_n} = \frac{\mathrm{elecost}^{\mathrm{high}}}{\mathrm{elecost}_{d_i}}. \tag{6}$$

Our main objectives in this research are to map tasks suitably to virtual resources while minimizing makespan and SLA violations. Therefore, makespan is evaluated using the following equation:

$$\mathrm{ms}^k = \mathrm{ava}^n + e^k. \tag{7}$$

After calculation of makespan, our next objective is to calculate SLA violations. Primarily SLA violation depends upon active time of a host and performance degradation. Therefore, active time of a host and performance degradation calculated using the follwoing equations, respectively.

$$\mathrm{AT}_{H_i} = \frac{1}{p}\sum_{s=1}^{p}\frac{\mathrm{vio\ time}_{H_i}}{\mathrm{AT}_{H_i}}, \tag{8}$$

$$\mathrm{pe}_{dg} = \frac{1}{n}\sum_{a=1}^{n}\frac{pe_{dg}^{p}}{\mathrm{to}_{vm}^{p}}. \tag{9}$$

Using above equations (8) and (9) we calculate SLA violations as follows:

$$\mathrm{SLA}_{\mathrm{vio}} = \mathrm{AT}_{H_i} * \mathrm{Pe}_{dg}. \tag{10}$$

In this research, as discussed in abstract and introduction section, existing authors used various metaheuristic and nature-inspired approaches for scheduling in cloud paradigm. In this manuscript, we used whale optimization algorithm which can cover entire problem solution space either exploit or by shrinking mechanisms. To achieve this, we calculate the optimization function as follows:

$$f(x) = \min\Big(\sum \mathrm{ms}^k(x), \mathrm{SLA}_{\mathrm{vio}}(x)\Big). \tag{11}$$

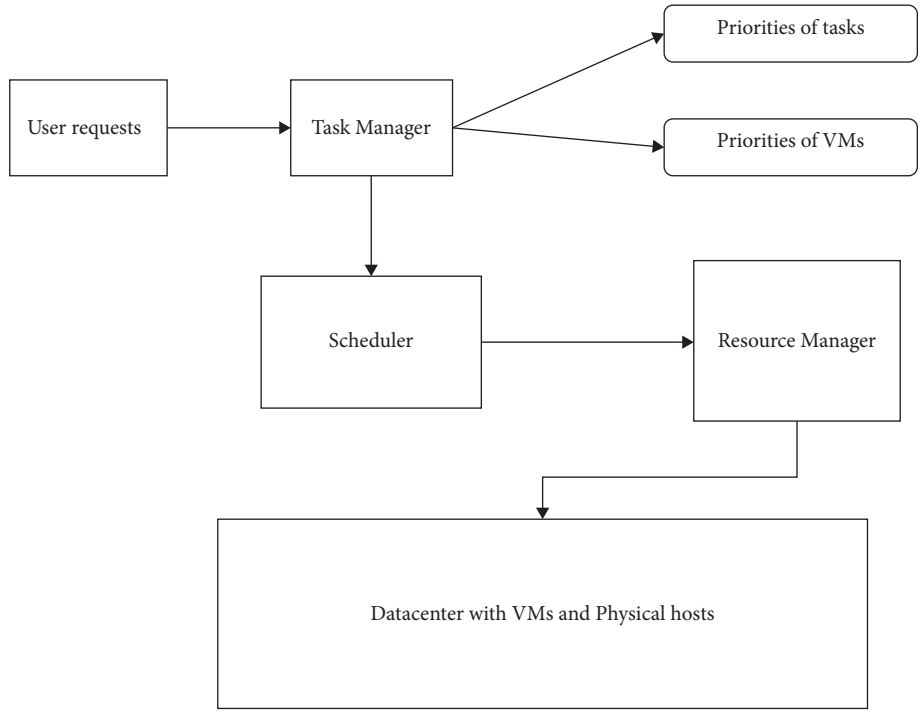


Figure 1: Proposed system architecture.

After carefully formulated mathematical modeling, in the next section we discussed about methodology used in our scheduling mechanism.

## 4. Methodology and Proposed Task-Scheduling Mechanism Using Whale Optimization

Whale optimization [21] used as methodology for scheduling in Cloud Computing in this research. It is a nature-inspired approach, which is based on whale's behavior in the nature. This algorithm initially starts with whale population, i.e., agents and looks for prey until its best solution is identified. It continuously searches for prey with these agents by using two phases, exploitation or shrinking phase. In this algorithm, a humpback whale population generated randomly and it is represented as $Q_a(b = 1, 2, \ldots . K)$ and best search agent is represented as $Q^*$. Initially position of prey is identified randomly and assumed that current prey indicated as best solution identified by search agent. Current best solution identified can be represented as

$$\overrightarrow{S} = \left| \overrightarrow{M} . \overrightarrow{Q}^* (x) - \overrightarrow{Q} (x) \right|, \tag{12}$$

where $x$ represents present iteration, $\overrightarrow{Q}(x)$ represents position vector, $\overrightarrow{Q}^*(x)$ represents best solution for identified position vector, and $\overrightarrow{M}$ represents coefficient vector.

Updated next position of search agent is calculated as follows:

$$\overrightarrow{Q} (x + 1) = \overrightarrow{Q}^* (x) - \overrightarrow{E} . \overrightarrow{S}, \tag{13}$$

where $\overrightarrow{E}$ represents coefficient vector. $\overrightarrow{E}$, $\overrightarrow{M}$ are calculated using the following equations:

$$\overrightarrow{E} = 2 \overrightarrow{u} . \overrightarrow{w} - \overrightarrow{u}, \tag{14}$$

$$\overrightarrow{M} = 2 . \overrightarrow{w}. \tag{15}$$

Next best search agent is identified based on modification of values of coefficient vectors. Initially, value of $\overrightarrow{u}$ ranges from 2 to 0. Value of $\overrightarrow{w}$ indicates random number between 0 and 1. These are modified and when looking for next best search agent a phase named as exploitation begins where it consists of encircling and spiral updation mentioned in [22]. Values of coefficient vectors updated to new values as [−1, 1] and then for this spiral updation position of agent is calculated using the following equation:

$$\overrightarrow{Q} (x + 1) = \overrightarrow{s^f} . b^{ce} . \cos (2\pi r) + \overrightarrow{Q}^* (x), \tag{16}$$

where $c$ is a constant and $e$ is a value that lies in between interval −1 and 1. $\overrightarrow{s^f}$ is represented as follows:

$$\overrightarrow{s^f} = \left| \overrightarrow{Q}^* (x) - \overrightarrow{Q} (x) \right|. \tag{17}$$

Entire search agent process is carried out between two phases, i.e., either shrinking or exploitation based on chosen probability of a search agent. If probability is less than 0.5 it uses shrinking approach , which is calculated by using equation (13). Otherwise, if probability is greater than or equals to 0.5 it uses spiral updation. After exploitation phase, exploration starts using an agent by choosing it randomly. It is represented by using the following equations:

**Input:** set of tasks $t_k = \{t_1, t_2, t_3, \ldots\ldots t_k\}$, set of VMs $vm_n = \{vm_1, vm_2, \ldots\ldots.., vm_n\}$, set of hosts $H_i = \{H_1, H_2, H_3, \ldots.., H_i\}$, Set of datacenters $D_j = \{D_1, D_2, \ldots\ldots, D_j\}$
**Output:** Scheduling of tasks mapped to VMs while minimizing makespan, SLA Violation.
Start
Populate whale population randomly
Calculation of priorities of $t_k$ using equation (5).
Calculation of priorities of $vm_n$ using equation (6).
Calculation of fitness function using equation (11).
if (probability < 0.5)
if ($|E| < 1$)
calculate search agent position using equation (13).
else if ($|E| < 1$)
calculate search agent position using equation (19).
End if
End if
If (probability ≥ 0.5)
Calculate position of search agent using equation (16).
End if
Calculate makespan, SLA violation using equations (7) and (10).
If (agent moves outside region)
Calculate $\overrightarrow{Q}^*$
Update iterations
End if
End

Algorithm 1: Proposed task-scheduling algorithm using whale optimization.

$$\overrightarrow{S} = \left|\overrightarrow{M}.\overrightarrow{Q}^{\text{rand}} - \overrightarrow{Q}\right|, \tag{18}$$

$$\overrightarrow{Q}(x+1) = \overrightarrow{Q}^{\text{rand}}(x) - \overrightarrow{E}.\overrightarrow{S}. \tag{19}$$

Termination begins after exploration after search agent moves out of region and this process continues till best solution is arrived is shown in Algorithm 1.

## 5. Simulation Setup and Results

This section discusses clear configuration settings for simulation and simulation results.

*5.1. Simulation Configuration Settings.* In our research, simulation was carried out using CloudSim [6] toolkit. This simulator helps us to simulate exact simulated environment for cloud paradigm. It was installed upon a machine which consists of configuration of 16 GB RAM, 5 TB hard disk, i7 processor. Table 3 represents exact standard configuration settings used in our simulation.

*5.2. Calculation of Makespan.* Initially, we calculated makespan using random generated workload and after that we used real time workload, i.e., BigDataBench workload [23]. We compared our proposed whale approach against existing PSO, ACO, GA, and W-Scheduler algorithms.

Table 4 represents the calculation of makespan for PSO, ACO, and proposed whale scheduler for 100, 500, and 1000 tasks. Makespan generated with random generated workload for PSO is 1289.5, 1678.76, and 1989.56, respectively. Makespan generated with random generated workload for ACO is 1156.9, 1563.8, and 2146.8, respectively. Makespan generated with random generated workload for GA is 1543.8, 1475.3, and 1934.57, respectively. Makespan generated with random generated workload for W-scheduler is 1058.35, 1342.78, and 1864.9, respectively. Makespan generated with random generated workload for proposed whale scheduler is 968.9, 1257.9, and 1784.8, respectively.

Table 3: Simulation configuration settings.

| Entity name | Quantity |
|---|---|
| No. of tasks | 100–1000 |
| Length of tasks | 780,000 |
| Memory of host | 16 GB |
| Storage capacity of host | 5 TB |
| Bandwidth of network | 1000 Mbps |
| No. of virtual machines | 20 |
| Memory of virtual machine | 1024 MB |
| Virtual machine bandwidth | 5 Mbps |
| Virtual machine monitor | Xen |
| Operating system | Linux |
| Number of datacenters | 5 |

Table 4: Calculation of makespan using a random generated workload.

| Tasks | PSO | ACO | GA | W-scheduler | Proposed whale scheduler |
|---|---|---|---|---|---|
| 100 | 1289.5 | 1156.9 | 1543.8 | 1058.35 | 968.9 |
| 500 | 1678.76 | 1563.8 | 1475.3 | 1342.78 | 1257.9 |
| 1000 | 1989.56 | 2146.8 | 1934.57 | 1864.9 | 1784.8 |

Table 5: Calculation of makespan using BigDataBench workloads.

| Tasks | PSO | ACO | GA | W-scheduler | Proposed whale scheduler |
|---|---|---|---|---|---|
| 100 | 1367.8 | 1243.9 | 1437.7 | 1138.3 | 1036.9 |
| 500 | 1747.9 | 1643.9 | 1532.9 | 1476.4 | 1387.9 |
| 1000 | 2045.7 | 2387.8 | 2243.6 | 1956.56 | 1899.5 |

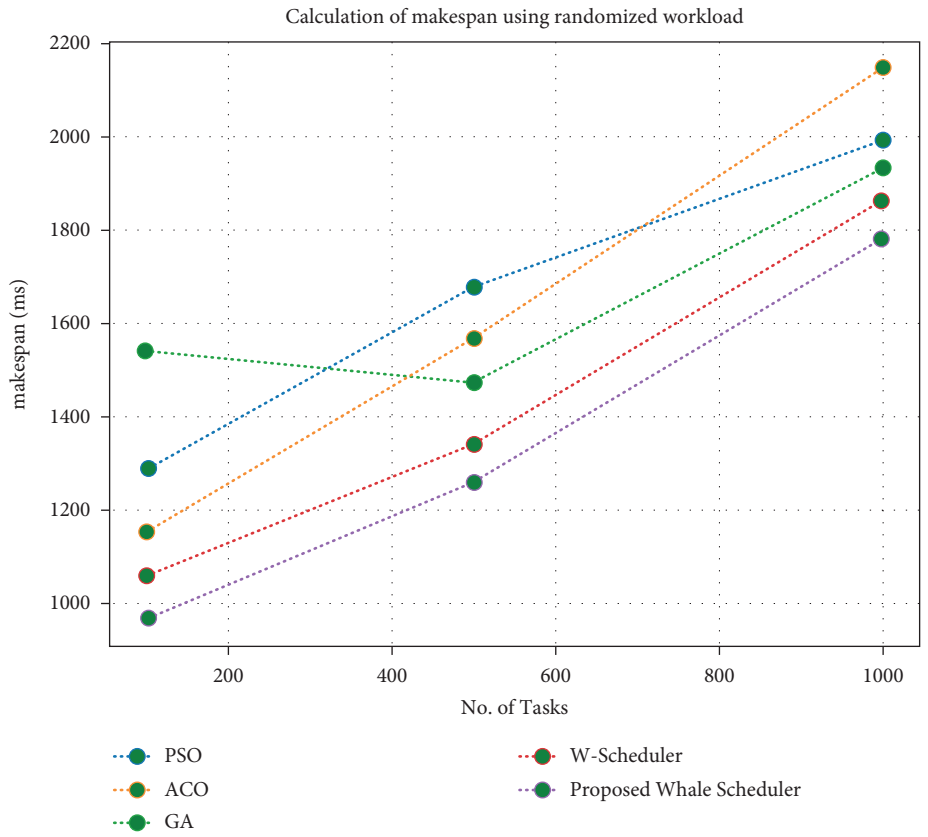


Figure 2: Calculation of makespan using a random workload.

Table 5 represents calculation of the makespan for PSO, ACO, and proposed whale scheduler for 100, 500, and 1000 tasks. Makespan generated with BigDataBench workload for PSO is 1367.8, 1747.9, and 2045.7, respectively. Makespan generated with BigDataBench workload for ACO is 1243.9, 1643.9, and 2387.8, respectively. Makespan generated with BigDataBench workload for GA is 1437.7, 1532.9, and 2243.6, respectively. Makespan generated with BigDataBench workload for W-scheduler is 1138.3, 1476.4, and 1956.56, respectively. Makespan generated with BigDataBench workload for proposed whale scheduler is 1036.9, 1387.8, and 1899.5, respectively.

From Figures 2 and 3 we can clearly observe that our proposed whale scheduler improves makespan over state-of-the art algorithms. The reason for the improvement of makespan over existing algorithms is because our proposed whale scheduler calculates priorities of tasks and VMs for all sets of tasks coming onto cloud console and whale approach carefully schedule tasks based on these priorities which minimizes makespan as mentioned in Figures 2 and 3.

*5.3. Calculation of SLA Violation.* Table 6 represents calculation of SLA violation for PSO, ACO, and proposed whale scheduler for 100, 500, and 1000 tasks. SLA violation generated with random generated workload for PSO is 17, 25, and 28, respectively. SLA violation generated with random generated workload for ACO is 12, 18, and 22, respectively. SLA violation generated with random generated workload for GA is 15, 12, and 21, respectively. SLA violation generated with random generated workload for W-Scheduler scheduler is 12, 10, and 9, respectively. SLA violation generated with random generated workload for proposed whale scheduler is 5, 9, and 18, respectively.

Table 7 represents calculation of SLA violation for PSO, ACO, and proposed whale scheduler for 100, 500, and 1000 tasks. SLA violation generated with random generated workload for PSO is 19, 30, and 35, respectively. SLA violation generated with random generated workload for ACO is 10, 12, and 18, respectively. SLA violation generated with random generated workload for GA is 18, 21, and 29, respectively. SLA violation generated with random generated

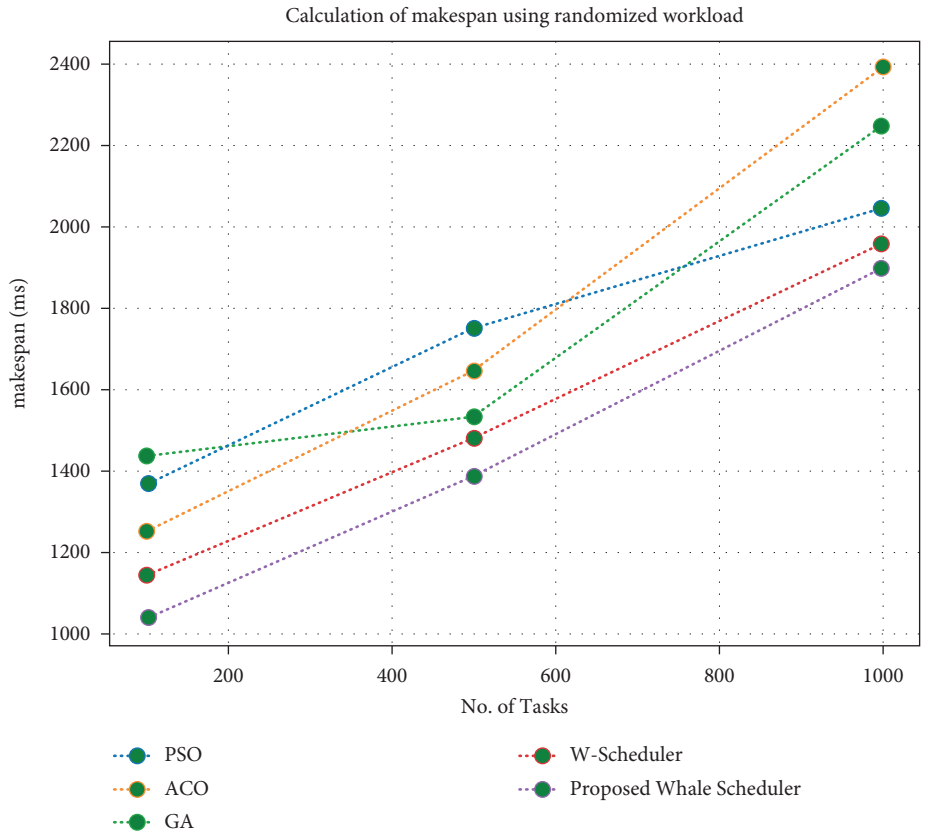


Figure 3: Calculation of makespan using the BigDataBench workloads.

Table 6: Calculation of SLA violation using random generated workload.

| Tasks | PSO | ACO | GA | W-scheduler | Proposed whale scheduler |
|---|---|---|---|---|---|
| 100 | 17 | 12 | 15 | 12 | 5 |
| 500 | 25 | 18 | 12 | 10 | 9 |
| 1000 | 28 | 22 | 21 | 19 | 18 |

Table 7: Calculation of SLA violation using a BigDataBench workload.

| Tasks | PSO | ACO | GA | W-scheduler | Proposed whale scheduler |
|---|---|---|---|---|---|
| 100 | 19 | 10 | 18 | 17 | 8 |
| 500 | 30 | 12 | 21 | 19 | 10 |
| 1000 | 35 | 18 | 29 | 15 | 12 |

workload for W-scheduler is 17, 19, and 10, respectively. SLA violation generated with random generated workload for proposed whale scheduler is 8, 10, and 12, respectively.

From Figures 4 and 5 we can clearly observe that our proposed whale scheduler minimizes SLA violations over state-of-the-art algorithms. The reason for the minimization of SLA violation over existing algorithms is that our proposed whale scheduler calculates priorities of tasks and VMs for all sets of tasks coming onto cloud console and whale approach carefully schedule tasks based on these priorities which minimizes makespan as mentioned in Figures 4 and 5.

*5.4. Analysis of Results.* This subsection gives detailed analysis on results obtained through our proposed whale scheduler. Initially for our scheduling algorithm, we have given randomized workload and evaluated over existing algorithms. After the initial evaluation we have given the workload from BigDataBench workloads [23]. For both the workloads we ran simulation with 100, 500, and 1000 tasks with 50 iterations. Tables 8 and 9 represents improvement percentage of makespan and SLA violations over existing baseline approaches while evaluating with proposed whale scheduler.

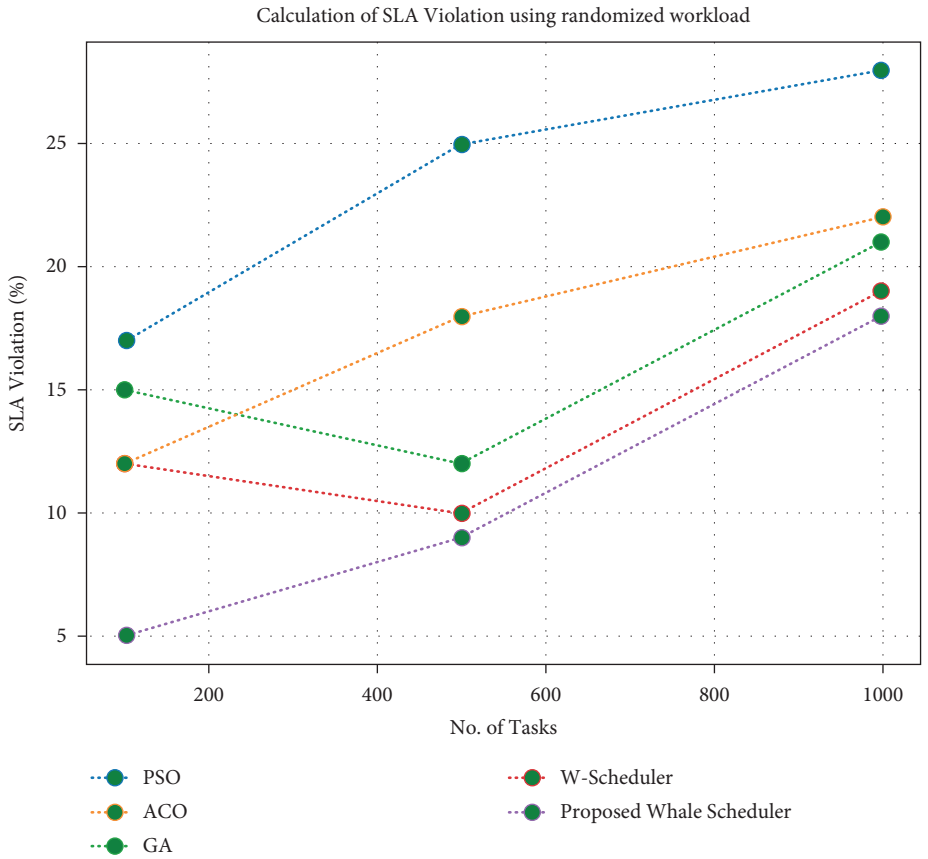


Figure 4: Calculation of SLA violation using the random workload.

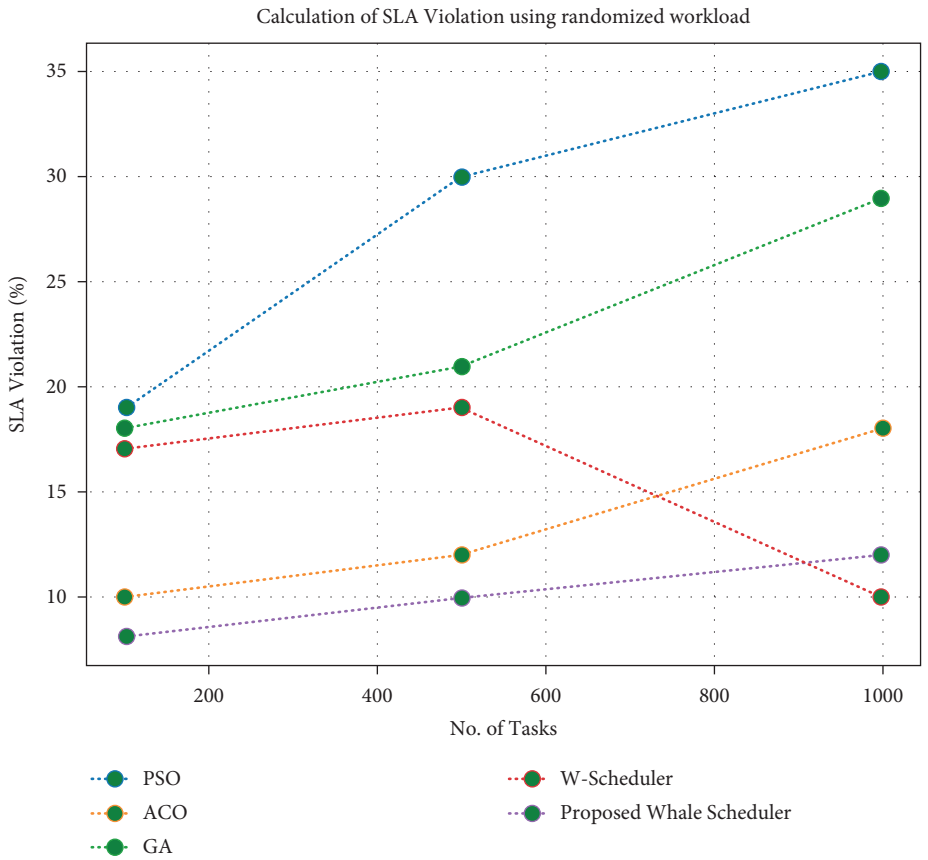


Figure 5: Calculation of SLA violation using the BigDataBench workload.

Table 8: Improvement of makespan over existing algorithms with various workloads.

| | Improvement of makespan over existing algorithms | | | |
|---|---|---|---|---|
| Type of workload/algorithm | PSO (%) | ACO (%) | GA (%) | W-scheduler (%) |
| Randomized | 20.07 | 17.55 | 19.9 | 6.35 |
| BigDataBench | 17.3 | 17.86 | 17.64 | 5.93 |

Table 9: Improvement of SLA violations over existing algorithms with various workloads.

| | Improvement of SLA violations over existing algorithms | | | |
|---|---|---|---|---|
| Type of workload/algorithm | PSO (%) | ACO (%) | GA (%) | W-scheduler (%) |
| Randomized | 56.76 | 42.17 | 35.29 | 24.53 |
| BigDataBench | 63.42 | 23.33 | 55.51 | 40.1 |

From Tables 8 and 9, it is evident that our proposed whale scheduler improvised makespan and SLA violations carefully over existing state-of-the-art algorithms.

## 6. Conclusion and Future Work

Task scheduling in Cloud Computing poses challenges to cloud provider as workload coming to cloud console is diversified and heterogeneous. It is challenging for cloud users as well if a proper scheduler is not employed in cloud paradigm. It leads to increase in makespan and SLA violations, which affects the quality of service. Therefore, in this manuscript, we proposed an effective task-scheduling algorithm, which takes priorities of tasks and VMs to appropriately map incoming tasks to virtual resources. To model this scheduling approach we used a whale optimization algorithm. Entire simulation and experiments are implemented on CloudSim. It is compared against existing algorithms PSO, ACO, GA, and W-scheduler. Initially in simulation random, generated workload was used and later a real time dataset was used, i.e., BigDataBench. From results, it evident that our proposed whale scheduler outperforms existing approaches by minimizing makespan and SLA violations. From simulations we observed that proposed whale scheduler improved makespan for PSO, ACO, GA, and W-schedulers by 20.07%, 17.55%, 19.9%, and 6.35%, respectively, and 17.3%, 17.86%, 17.64%, and 5.93%, respectively, for BigDataBench orkloads. SLA violations improved over PSO, ACO, GA, and W-Scheduler by 56.76%, 42.17%, 35.29%, and 24.53%, respectively, and 63.42%, 23.33%, 55.51%, and 40.1%, respectively for BigDataBench workloads. In the future, we need to employ a machine learning approach for better handling of scheduling algorithm in Cloud Computing paradigm.

## Data Availability

No data used for supporting this study were disclosed by the authors.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References

[1] D. Sabella, "Principles of edge computing, fog and cloud computing," *Multi-access Edge Computing: Software Development at the Network Edge*, pp. 3–18, Springer, Heidelberg, Germany, 2021.

[2] M. U. Bokhari, Q. Makki, and Y. Kord Tamandani, "A survey on cloud computing," *Big Data Analytics*, pp. 149–164, Springer, Singapore, 2018.

[3] I. Ahmed, "A brief review: security issues in cloud computing and their solutions," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 17, no. 6, pp. 2812–2817, 2019.

[4] M. S. Sudheer and M. Vamsi Krishna, "Dynamic PSO for task scheduling optimization in cloud computing," *International Journal of Recent Technology and Engineering*, vol. 74, 2019.

[5] A. M. Senthil Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Personal Communications*, vol. 107, pp. 1835–1848, 2019.

[6] S. Badr, E. M. Ahmed, G. Attiya, and A. A. Nasr, "Task consolidation based power consumption minimization in cloud computing environment," *Multimedia Tools and Applications*, vol. 353, pp. 1–29, 2022.

[7] S. Nabi, M. Ahmad, M. Ibrahim, and H. Hamam, "AdPSO: adaptive PSO-based task scheduling approach for cloud computing," *Sensors*, vol. 22, p. 920, 2022.

[8] M. Ibrahim, S. Nabi, A. Baz et al., "An in-depth empirical investigation of state-of-the-art scheduling approaches for cloud computing," *IEEE Access*, vol. 8, pp. 128282–128294, 2020.

[9] M. Nanjappan and P. Albert, "Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment," *Concurrency and Computation: Practice and Experience*, vol. 34, p. 5517, 2022.

[10] S. Nabi and M. Ahmed, "PSO-RDAL: particle swarm optimization-based resource-and deadline-aware dynamic load balancer for deadline constrained cloud tasks," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 4624–4654, 2022.

[11] K. Dubey and S. C. Sharma, "A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 32, Article ID 100605, 2021.

[12] K. Dubey and S. C. Sharma, "A hybrid multi-faceted task scheduling algorithm for cloud computing environment,"

*International Journal of System Assurance Engineering and Management*, vol. 207, pp. 1–15, 2021.

[13] P. Krishnadoss, N. Pradeep, J. Ali, M. Nanjappan, P. Krishnamoorthy, and V. Kedalu Poornachary, "CCSA: hybrid cuckoo crow search algorithm for task scheduling in cloud computing," *International Journal of Intelligent Engineering and Systems*, vol. 14, pp. 241–250, 2021.

[14] E. Elsedimy and F. Algarni, "MOTS-ACO: an improved ant colony optimiser for multi-objective task scheduling optimisation problem in cloud data centres," *IET Networks*, vol. 11, no. 2, pp. 43–57, 2022.

[15] 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.

[16] M. Hussain, L. F. Wei, A. Lakhan, S. Wali, S. Ali, and A. Hussain, "Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing," *Sustainable Computing: Informatics and Systems*, vol. 30, Article ID 100517, 2021.

[17] L. Abualigah and M. Alkhrabsheh, "Amended hybrid multiverse optimizer with genetic algorithm for solving task scheduling problem in cloud computing," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 740–765, 2022.

[18] T. Jena and J. R. Mohanty, "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing," *Arabian Journal for Science and Engineering*, vol. 43, pp. 4115–4130, 2018.

[19] Z. Zhou, F. Li, H. Zhu, H. Xie, J. H. Abawajy, and M. U. Chowdhury, "An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments," *Neural Computing & Applications*, vol. 32, no. 6, pp. 1531–1541, 2020.

[20] P. Krishnadoss and P. Jacob, "OCSA: task scheduling algorithm in cloud computing environment," *International Journal of Intelligent Engineering and Systems*, vol. 11, pp. 271–279, 2018.

[21] G. Natesan and A. Chokkalingam, "An improved grey wolf optimization algorithm based task scheduling in cloud computing environment," *The International Arab Journal of Information Technology*, vol. 17, no. 1, pp. 73–81, 2020.

[22] P. S. Rawat, P. Dimri, P. Gupta, and G. P. Saroha, "Resource provisioning in scalable cloud using bio-inspired artificial neural network model," *Applied Soft Computing*, vol. 99, pp. 1–31, 2021.

[23] L. Wang, J. Zhan, C. Luo et al., "Bigdatabench: a big data benchmark suite from internet services," in *Proceedings of the 2014 IEEE 20th international symposium on high performance computer architecture (HPCA)*, pp. 488–499, IEEE, Orlando, FL, USA, February, 2014.

[24] Z. Yu, "Research on optimization strategy of task scheduling software based on genetic algorithm in cloud computing environment," *Wireless Communications and Mobile Computing*, vol. 2022, Article ID 3382273, 9 pages, 2022.

[25] L. W. Jia, K. Li, and X. Shi, "Cloud computing task scheduling model based on improved whale optimization algorithm," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 4888154, 13 pages, 2021.

[26] R. Masadeh, N. Alsharman, A. Sharieh, B. A. Mahafzah, and A. Abdulrahman, "Task scheduling on cloud computing based on sea lion optimization algorithm," *International Journal of Web Information Systems*, vol. 17, no. 2, pp. 99–116, 2021.

[27] L. Ni, X. Sun, X. Li, and J. Zhang, "GCWOAS2: multiobjective task scheduling strategy based on Gaussian cloud-whale optimization in cloud computing," *Computational Intelligence and Neuroscience*, vol. 2021, Article ID 5546758, 17 pages, 2021.

[28] S. P. Praveen, H. Ghasempoor, N. Shahabi, and F. Izanloo, "A hybrid gravitational emulation local search-based algorithm for task scheduling in cloud computing," *Mathematical Problems in Engineering*, vol. 2023, Article ID 6516482, 9 pages, 2023.

[29] L. Shi, J. Xu, L. Wang et al., "Multijob associated task scheduling for cloud computing based on task duplication and insertion," *Wireless Communications and Mobile Computing*, vol. 2021, Article ID 6631752, 13 pages, 2021.