Computers, Materials & Continua
DOI: 10.32604/cmc.2023.031614
Article

Improvised Seagull Optimization Algorithm for Scheduling Tasks in Heterogeneous Cloud Environment

Pradeep Krishnadoss*, Vijayakumar Kedalu Poornachary, Parkavi Krishnamoorthy and Leninisha Shanmugam

Vellore Institute of Technology, Chennai, 632014, India
*Corresponding Author: Pradeep Krishnadoss. Email: pradeep.k@vit.ac.in
Received: 22 April 2022; Accepted: 23 June 2022

Abstract: Well organized datacentres with interconnected servers constitute the cloud computing infrastructure. User requests are submitted through an interface to these servers that provide service to them in an on-demand basis. The scientific applications that get executed at cloud by making use of the heterogeneous resources being allocated to them in a dynamic manner are grouped under NP hard problem category. Task scheduling in cloud poses numerous challenges impacting the cloud performance. If not handled properly, user satisfaction becomes questionable. More recently researchers had come up with meta-heuristic type of solutions for enriching the task scheduling activity in the cloud environment. The prime aim of task scheduling is to utilize the resources available in an optimal manner and reduce the time span of task execution. An improvised seagull optimization algorithm which combines the features of the Cuckoo search (CS) and seagull optimization algorithm (SOA) had been proposed in this work to enhance the performance of the scheduling activity inside the cloud computing environment. The proposed algorithm aims to minimize the cost and time parameters that are spent during task scheduling in the heterogeneous cloud environment. Performance evaluation of the proposed algorithm had been performed using the Cloudsim 3.0 toolkit by comparing it with Multi objective-Ant Colony Optimization (MO-ACO), ACO and Min-Min algorithms. The proposed SOA-CS technique had produced an improvement of 1.06%, 4.2%, and 2.4% for makespan and had reduced the overall cost to the extent of 1.74%, 3.93% and 2.77% when compared with PSO, ACO, IDEA algorithms respectively when 300 vms are considered. The comparative simulation results obtained had shown that the proposed improvised seagull optimization algorithm fares better than other contemporaries.

Keywords: Cloud computing; task scheduling; cuckoo search (CS); seagull optimization algorithm (SOA)

1 Introduction

Resources that constitute processors, storage, memory and executable applications present in cloud could be altered and fit dynamically according to demand imposed by the received user application. Resource acquisition in this dynamic manner has made the cloud immensely popular among users balancing their economic needs. This could be attributed to the scalable nature of cloud resources. Cloud resources are shared dynamically through virtualization concept in which the capability of multiple virtual systems is multiplied and represented as a single remote physical server. Resource usage in cloud gets optimized through the virtualization concept. Virtual Machine (VM) constitutes the core section of the cloud datacentre.

On demand facilities offered by cloud over the Internet include storage, servers, databases and software applications. Cloud computing is an important off-shoot of the distributed computing technology Kalra et al. [1]. The user submitted tasks through the task manager module get executed at the cloud. After analysing the tasks thoroughly, the task scheduler maps them with the available resources in a dynamic manner. Mapping the tasks with the available resources is carried out with the assistance of resource information server. Finally, the tasks are allocated to the virtual machines using scheduling algorithms. Any scheduling approach is considered efficient provided, it makes use of the available resources in an optimized manner.

Efficacy of the task scheduling activity determines the cloud performance. Shared resources of cloud are made use of by the application developers. Highly efficient algorithms are needed for mapping the available cloud resources with the requirements demanded by the user tasks. Complex but, efficient algorithms are the need of the hour for providing the users with efficient solutions. Demand imposed by simple applications are more likely to be easily manageable than those imposed by complex applications. Such complex applications need efficient algorithms to effectively managing the datacentres Zuo et al. [2].

NP-hard problems are usually solved through meta-heuristic approaches. In meta-heuristic approach, the task scheduling activity is articulated as an optimization issue. Even though optimal solutions are not always generated through meta-heuristics, they assist in producing near optimal solutions with minimal complexity. Most of the results generated are highly encouraging and sync well. These approaches get cornered themselves inside a local optima leading to their performance degradation Buyya et al. [3], Nanjappan et al. [4].

In this proposed work, a hybridized approach based on Seagull Optimization Algorithm (SOA) and Cuckoo search (CS) for optimizing the task scheduling had been presented. It minimizes the task execution time and cost of the resources. The contributions of this research work are summarized as follows:

1) Several literatures had been reviewed for analysing the overhead imposed by scheduling techniques on the performance of cloud environment.
2) An innovative, hybrid meta-heuristic algorithm had been designed by integrating SOA and CS algorithms for addressing the challenges and requirements of task scheduling. The proposed hybridized algorithm effectively minimizes the makespan and cost parameters leading to improved performance of the cloud environment.
3) Extensive experiment simulations had been done for assessing the performance of the proposed technique by comparing it with other existing techniques.

The remainder of this article is organized as follows. The study of literature review related to scheduling had been presented in Section 2. The solution framework and problem description had

been described in Section 3. In Section 4, the proposed hybrid SOA-CS technique had been presented in detail. In Section 5, the experimental evaluation and discussions are reported and the Section 6 depicts the conclusion remarks.

2 Related Works

Dubey et al. [5] had observed that the scheduling of various cloudlets having intrinsic deadline constraints against the cloud resources and subsequently satisfying the QoS requirements is highly demanding. The authors had addressed the task scheduling issue in cloud by proposing a hybrid Chemical Reaction Partial Swarm Optimization algorithm to allot several independent tasks on the available virtual machines at any particular instant. The chemical reaction optimization algorithm had been integrated with the particle swarm optimization algorithm. The optimal schedule sequence for processing the tasks had been generated based on demand and deadline constraints at any instant for improvising the quality features like cost, energy and makespan.

Krishnadoss et al. [6] had proposed a hybrid Cuckoo Crow Search Algorithm (CCSA) for carrying out task scheduling activity in the cloud. Parasitic features of the cuckoo bird, along with the food collecting behaviour of crow are imitated in the proposed algorithm. Crows usually turn focus on food possessed by other neighbour crows, thinking that neighbours have better food than what they have. This thinking propels them to snatch the food from their neighbours. The proposed CCSA had been inspired from such bird behaviours and designed accordingly to select a more viable virtual machine where the task scheduling activity could be efficiently carried out in the cloud environment. The proposed CCSA had been compared with Multi objective-Ant Colony Optimization (MO-ACO), Ant Colony Optimization (ACO) algorithm and Min-Min algorithm and had accounted for reduced makespan and cost parameters.

Alsadie [7] had suggested that scheduling in cloud could be done by allocating the tasks to suitable virtual machine datacentres resulting in minimized expenditure of energy, makespan and cost parameters. The authors had applied this strategy in their Meta-heuristic framework that Dynamically allocates the Virtual Machines (MDVMA) for carrying out an optimized scheduling process in the cloud. A multi-objective scheduling strategy had been implemented in the proposed MDVMA by making use of the Non-dominated Sorting Genetic Algorithm (NSGA-II) for optimizing the task scheduling process. The proposed model effectively minimizes the energy expended, makespan and cost parameters to satisfy the cloud service providers based on their requirements.

Gao et al. [8] had categorized the scheduling of workflow in cloud as a NP-hard problem. The authors had proposed a novel strategy for scheduling the multi-objective workflow in cloud by limiting the number of its instances and providing a scalable mixture of such instances. The authors had designed their algorithm by hybridizing the genetic algorithm and artificial bee colony optimization algorithm. The decoding heuristics had been applied for scheduling the tasks with available resources to minimize the makespan and cost parameters. The authors had analysed the performance of their proposed algorithm through simulated experiments and had validated it against real world scientific applications.

Albert et al. [9] had categorized the assignment of tasks to appropriate virtual machines as a tricky process due to which scheduling is generally categorized as a NP-hard problem. Cloud errand booking had been tried out with the help of several met-heuristic solutions. The authors had designed a novel hybridized algorithm named Whale Harmony Optimization Algorithm (WHOA) by integrating the Whale Optimization Algorithm and Harmony Search Algorithm. The proposed WHOA had been

applied for regulating the framework stack and efficiently minimize the makespan and cost parameters during the task scheduling process in heterogeneous cloud environment.

Mangalampalli et al. [10] had suggested that the virtual resources present in the cloud needs to be intelligently provisioned for optimizing the entire scheduling activity. This could be realized by minimizing the makespan and maximizing the utilization of resources present in the cloud. The authors had considered the energy expended, migration time and total power cost factors in datacentres for further improvising the cloud task scheduling activity. In this regard, the authors had proposed the Cat Swarm Optimization algorithm that focuses on minimizing these parameters at the datacentres. Tasks and virtual machines could be affixed with priority during scheduling at the task level and virtual machine level respectively. This assists in efficient scheduling by enabling valid mapping of tasks to appropriate virtual machines.

Zuo et al. [11] had proposed a multi-objective scheduling technique by considering the resource-cost model. The authors had taken into account of the makespan and cost parameters as constraints for improvising the performance of task scheduling activity in cloud environment. The authors had proposed an improvised Ant Colony Optimization algorithm. They had set up two constraint functions for evaluating and providing feedback with respect to makespan and budget cost simultaneously. The feedback function incorporated in the algorithm design helped to make timely quality adjustments while devising optimal solution. The authors had evaluated their proposed algorithm with respect to makespan, cost, deadline violation and resource utilization parameters.

Somasundaram et al. [12] had designed an efficient CLOUD Resource Broker (CLOUDRB) that manages the available cloud resources in an effective manner to complete tasks related to scientific applications such that user specified deadline constraints are upheld. The authors had designed their model by integrating the Deadline based Job Scheduling strategy with the Particle Swarm Optimization algorithm for efficiently allocating the resources among the received tasks. A fitness function centred around minimization of makespan and cost had been designed to meet the specified requirements. Simulations were carried out in the Matlab programming environment by modelling the high-performance computational jobs and cloud resources. The proposed approach remains effective in minimizing the makespan, cost, job rejection ratio and maximizing the total jobs that get completed within the specified time limits.

Yao et al. [13] had proposed the ECMSMOO technique by considering multiple swarms in which each of these swarms are engaged with an improvised multi-objective particle swarm optimization strategy for deducing non-dominant solutions with a single primary objective. To escape from being trapped into local optima, the authors had embedded an endocrine stirred mechanism during the particle evolution stage. The proposed ECMSMOO has a competitive and cooperative technique infused among the swarms that effectively improves its performance. The performance efficiency of the proposed work had been compared with other existing algorithms by submitting jobs falling under hybrid and parallel workflow category.

Sreenu et al. [14] had proposed the W-Scheduler by integrating a multi-objective model with the Whale Optimization Algorithm (WOA). The multi-objective model deduces the initial fitness value by taking into account of the cost factor of the CPU and memory. Final fitness value is then deduced by adding up the makespan and the cost value obtained. Results obtained had shown that the proposed model schedules the tasks to the virtual machines optimally by minimizing the makespan and cost parameters.

Table 1: Comparative table for literatures review

Author	Algorithm	Parameter considered	Compared algorithm	Tools/ environment
Dubey et al. [5]	Chemical reaction optimization (CRO) and particle swarm optimization (PSO)	Cost, energy, and makespan	PSO , IPSO , PSO-COGENT and CRO	CloudSim
Krishnadoss et al. [6]	Cuckoo crow search algorithm (CCSA)	Makespan and cost	MO-ACO, ACO, Min-Min	CloudSim
Alsadie [7]	MDVMA	Energy, makespan and cost	Artificial bee colony (ABC) algorithm, whale optimization algorithm (WOA) and particle swarm optimization (PSO)	CloudSim
Gao et al. [8]	multi-objective workflow scheduling in IaaS clouds	Makespan and cost	HPSO, PBACO, MGA, MABC, and HGAABC	CloudSim
Albert et al. [9]	Whale optimization algorithm and armony search algorithm.	Execution time, cost and energy consumption	OGWO, WOA and HS	CloudSim
Mangalampalli et al. [10]	Cat swarm optimization algorithm	Makespan, energy and Cost	HPC2 N and NASA	CloudSim
Zuo et al. [11]	A multi-objective optimization scheduling method (PBACO)	Makespan, cost, deadline violation rate and resource utilization	FCFS,ACO,Min-Min	CloudSim
Somasundaram et al. [12]	CLOUD Resource broker (CLOUDRB)	Makespan and cost	ACO,GA and RBA	Matlab
Yao et al. [13]	ECMSMOO	Makespan, cost, and energy	MOHEFT CMPSO	Workflow sim
Sreenu et al. [14]	W-Scheduler	Makespan and cost	BACO, SLPSO-SA, and SPSO-SA	CloudSim

The above literature review does not provide near optimal solution for Quality of Service parameters likemakespan and costwhen considered together as shown in table Tab. 1. In this proposed work, cost and time parameters had been taken up as performance metrics to optimize the task and resource allocation activity in the cloud. A novel, hybridized Seagull Optimization Algorithm (SOA) and Cuckoo search (CS) had been proposed for minimizing the cost and time factors impacting the performance efficiency pertaining to the task scheduling and resource allocation activities in the cloud computing environment.

3 Problem Definition and Explanation

This section provides a simplified explanation about the system framework, tasks and the resources that had been designed and incorporated in this work. The Tab. 2 shown below provides the meaning of the parameters that had been considered.

Table 2: Notations used in the SOA-CS scheduler

Symbol	Definitions
PM	Denotes a physical machine
VM	Denotes a virtual machine
T_{ji}	Tasks i, $1 < j < K$
N_{ji}	Resources j, $1 < i < K$
α, β	Control parameters
CPU_{ji}	Total capacity of CPU
$Exceuction^{Timeji}$	Execution time on the j^{th} virtual machine that is included under the i^{th} physical machine
$Task^{Length}$	Length of tasks
$Processor^{Capacity}$	Capacity of the processor

Let N and K denote the resources $R = \{R_1, R_2, \ldots, Rj, \ldots, R_n\}$ and tasks $T = \{T_1, T_2, \ldots, T_i, \ldots \ldots, T_k\}$ present in the proposed system respectively. Also assume that the resources present in virtual form too be considered as cloud resources.

In Fig. 1 depicted above, it could be seen that the user tasks are aggregated, analysed and passed onto the scheduler. The scheduler then allocates the required resources for each task. The resource monitor keeps track of available resources in the cloud environment that includes storage, computing and bandwidth. Scheduling policy comprises of a set of rules and policies based upon which tasks are assigned to appropriate resources (CPU, memory, and bandwidth) such that optimum level of performance and resources utilization could be achieved. QoS includes a host of parameters including execution time, resource utilization, cost, energy, bandwidth etc.

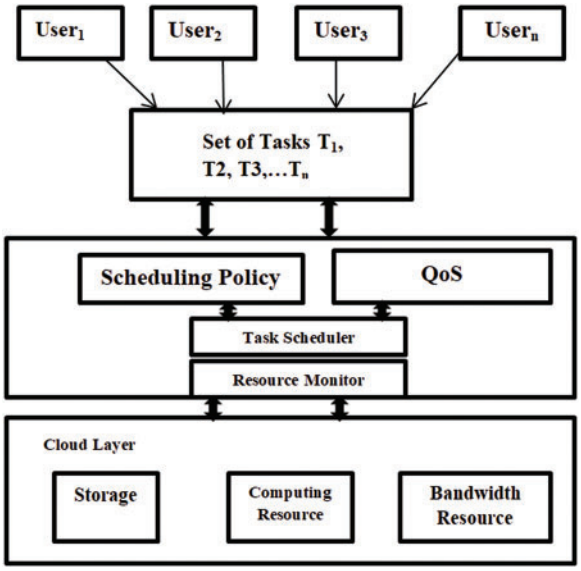


Figure 1: Framework for task scheduling in cloud

4 Proposed Hybrid SOA-CS

4.1 Cuckoo Search Algorithm

Yang et al. [15] had proposed the population based, nature driven, cuckoo search algorithm. The cuckoo birds have the habit of hatching their eggs in crow nests and other similar winged animal nests whenever they see the host birds' nests unoccupied. Let as assume that the host bird identifies the cuckoo birds' egg with probability $q_a \in [0, 1]$. Then host bird makes a decision either to drop cuckoo bird's eggs or to abandon its own nest.

This feature of cuckoo bird could be expressed mathematically. Let the total number of cuckoos present in the environment be denoted by C_{uck} and let everyone of them represent a nest. These individual nests could be regarded as one of the solutions for the optimization problem. The search space dimension be denoted by n. Then the vector hunt space for ith cuckoo at any time t could be represented as $F(t) = (e^1, e^d, \ldots, e^n)$ where $i = 1, 2, \ldots, C_{uck}$. Similarly, the next hunt space for the ith cuckoo at time $t + 1$ could be represented as:

$$F_i(t+1) = F_i(t) + a \otimes Levy(y) \tag{1}$$

where $a > 0$, is the step size which is a fixed constant and the arbitrary walk based on the Levy flight [15] is represented by $Levy(y)$. The constant a is fixed to a value of 1 and $\otimes$, denotes the entry-wise multiplication.

Using the Markov chain and Levy step, the arbitrary walk for next location could be deduced. By prolonging the Levy step, future lengths could be subsequently deduced. These Levy steps are deduced from Levy distribution that in turn are deduced from the Mantegna algorithm. The Levy step could be deduced as shown in equation below:

$$Levy(y) = u/|v| \frac{1}{Y-1} \tag{2}$$

where, the values of u and v are obtained through gaussian distribution and the value of y is selected between $1 < y < 3$. The standard deviations of u and v are depicted in equation given below:

$$\sigma_U(\gamma) = \left[\frac{\Gamma(1+\gamma) sin\left(\frac{\pi\gamma}{2}\right)}{\Gamma\left(\frac{1+\gamma}{2}\right)\gamma 2^{\left(\frac{\gamma-1}{2}\right)}} \right]^{\frac{1}{\gamma}} \quad and \ \sigma_v(\gamma) = 1 \tag{3}$$

For all those non-aggressive Levy steps, the $Levy(y)$ values are multiplied by a carefully selected factor β. $\Gamma(.)$ represents the Gamma function. The levy distributions in the case of larger advances are obtained from power law. Such distributions exhibit an infinite variance and the same is shown in equation given below:

$$Levy \sim u = t^{-\gamma} \tag{4}$$

Hence it could be seen that the CS algorithm produces lesser step sizes for shorter distances that represent intermittent noteworthy distance walking that gradually becomes longer over the increased time span.

4.2 Seagull Optimization Algorithm (SOA)

The Seagull Optimization Algorithm (SOA) had been inspired from the seagulls that are sea birds living in clusters, possessing migrating and hunting characteristics Das et al. [16]. Migration is a seasonal activity, where the seagulls move away in search of food. As any prey is spotted, the seagulls take a spiral type of motion to hunt down their prey. This activity of seagulls could be represented mathematically. Collisions between adjacent seagulls could be avoided by including an additional parameter M as shown below:

$$\vec{E}_c = MX\vec{E}_{current(x)}(x) \tag{5}$$

where, $\vec{E}_{current(x)}(x)$ indicates the present location of the search specialist during x^{th} iteration and M denotes the driving feature of the hunt agent taken into consideration. M is deduced as shown below:

$$M = e_c - \left(x' \left(\frac{e_c}{Max_{iteartion}} \right) \right) \tag{6}$$

where $x' = 0, 1, 2, \ldots, Max_{iteration}$ and f_c controls the value of M, acting as its boundary and gets reduced from e_c to 0 in a linear manner. In this present work, the estimated value of e_c had been fixed as 2.

Once the collisions are overcome, the search agents start moving in the direction of the location of the best seagull (regarded as ideal solution). This could be as shown below:

$$\vec{G}_s = NX\left(\vec{E}_{best}(x) - \vec{E}_{current}(x) \right) \tag{7}$$

In Eq. (7) shown above, $\vec{G}_s$ represents the candidate positions that are closer to the best-fit candidate $\vec{E}_{best}(x)$. The exploration and exploitation steps are balanced by assigning a random value to the coefficient N. The value of N could be deduced using equation shown below:

$$N = 2' \ M^{2'} \mathrm{X}\, rd \tag{8}$$

In Eq. (8), the value of *rd* is chosen arbitrarily from the range [0, 1].

Then, the candidates start updating their present location based on the optimal search agent by $\vec{D}_s$ quantity. The value of $\vec{D}_s$ could then be deduced as shown in equation below:

$$\vec{D}_s = \left| \vec{E}_s + \vec{G}_s \right| \tag{9}$$

Which indicates the actual lay-off between the chosen hunt specialist and the fittest of seagulls.

In the attacking step, the seagulls continuously vary their approach (based on speed) by changing their normal motion to that of a spiral form in air. Such behaviour of the seagulls could be represented in the x, y and z plane as shown in eqns. below:

$$x' = r \times \cos(k) \tag{10}$$

$$y' = r \times \sin(k) \tag{11}$$

$$Z' = r \times k \tag{12}$$

$$r = u \times e^{kv} \tag{13}$$

where, r denotes the measured radius of every spiral turn, k value is selected randomly from $[0 <= \mathrm{k} \leq 2]$. The twisting shape had been characterized and represented through the constants u and v, and e is fixed as base value of natural algorithm. Now Eqs. (9)–(13) are utilized in deducing the new position of the search specialist and the same had been depicted in equation given below:

$$\vec{E}_{current}(x) = \left(\vec{D}_s \times x' \times y' \times Z' \right) + \vec{E}_{best}(x) \tag{14}$$

In Eq. (14) shown above, $\vec{E}_{current}(x)$ retains the best results obtained thus far and accordingly revises the positions of other hunt specialists.

This extends the scope of the exploration phase making it more robust and capable of covering more search zones, without getting trapped within the local minima zone.

4.3 Hybrid SOA-CS

We had specified earlier that the number of cuckoos in the environment be represented as C, making a nest being made available for each cuckoo, where each nest signifies a solution. The Cuckoo Search algorithm makes use of the arbitrary Levy flight walkthrough that is assumed more robust while traversing the search space and derives the step size from Levy appropriation.

Once the progression of step is deduced, the Eq. (9) gets modified as shown in below equation:

$$\vec{D}_s = \left| \vec{E}_c + \vec{G}_s \times Levy\,(\gamma) \right| \tag{15}$$

$\vec{G}_s$ Represents best-fit candidate $\vec{E}_{best}(x)$, $\vec{D}_s$ represents optimal search agent by quantity, $Levy(\gamma)$ represents arbitrary Levy flight walkthrough.

This makes the exploration phase much stronger, covering several search zones, thereby breaking the clutches of getting trapped within the local minima, making it the most preferred strategy.

Once the next best particle gets available, it intelligently pursues the solution space to discover solution, moving ahead in a spiral manner. The projection components x' and y' shown in Eqs. (10) and (11) respectively get reduced continuously, with the altitude component z, together assist the hunting policy. The radius r in every spiral's motion makes the exploitation phase crispier. The spiral momentum adopted during the hunting of the prey is incorporated into hybridized approach. The Eq. (14) thus gets updated as shown in equation below:

$$\vec{E}_{current}(x) = \left(\vec{D}_s \times x' \times y' \times Z'\right) + \vec{E}_{best}(x) \tag{16}$$

where $\vec{E}_{current}(x)$ stores the best solution. Eq. (16) depicts the innovative preparation space for the proposed hybridized Seagull Optimization Algorithm-Cuckoo Search (SOA-CS), after the completion of a cycle. Algorithmic sequence of the proposed SOA-CS algorithm has been presented below.

4.4 Fitness Calculation

4.4.1 Fitness Parameters

The proposed work improvises the task scheduling activity in cloud by reducing the makespan and cost parameters. These parameters had been defined as below:

Makespan: It denotes the time expended in completing the execution of all tasks that are received. To achieve optimal solution, the value of makespan needs to be minimal. The makespan parameter is impacted by the length of task (TL) and the capacity of the processor (PC) as well. Task length denotes the total instructions that need to be executed as a part of each task. The execution time of the received tasks could be deduced using the following equation:

$$Execution^{Timeji} = \frac{Task^{Length}}{Processing^{Capacity}} \tag{17}$$

where TL ($Task^{Length}$) denotes the length of task and PC ($Task^{Length}$), the processing capability of the resource assigned.

Cost: The cost spent for a task can be calculated by taking into account of the number of virtual machine movements and dividing it by the total number of virtual machines that are assigned on any specific physical machine. The cost value could be deduced using the following equation:

$$C_i = \frac{1}{PM}\sum\nolimits_{i=1}^{m}\left(\frac{Number\ of\ movements\ in\ VM}{Total\ VMs}\right) \tag{18}$$

where Ci indicates cost and PM, the physical machine.

A fitness function needs to be designed using the makespan and cost parameters such that the task scheduling activity in cloud gets optimized. The fitness function designed should churn out near optimal solutions and the same has been depicted in the following equation:

$$\text{Fitness Function} = \sum\nolimits_{i=1}^{m} T_i\,(\alpha\ .\ \text{Makespan} + \beta.\text{Cost}) \tag{19}$$

where α, β are control parameters with their values in the range [0, 1].

The subsequent sub-sections depict the pseudocode, algorithm and flowchart of the proposed SOA-CS algorithm.

4.4.2 *Pseudocode: Proposed SOA-CS*

Input: Population $\vec{F}_{current}$

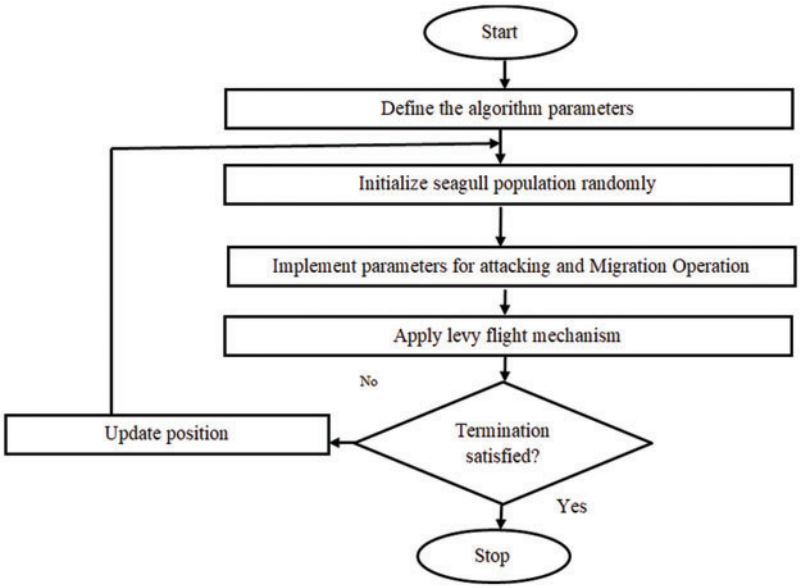
Output: Optimal search agent $\vec{F}_{best}$

1) Initialize the parameters A, B and $\text{Max}_{\text{iteration}}$
2) Set fc←2
3) Set u←1
4) Set v←1
5) While ($x < \text{Max}_{\text{iteration}}$) do
6) $\vec{E}_{best} \leftarrow$ ComputeFitness ($\vec{F}_{current}$)
7) rd←Rand(0, 1)
8) k←Rand(0, 2π)
9) r←$u \times e^{kv}$
10) Calculate the distance D → s
11) P←$x' \times y' \times z'$
12) $\vec{E}_{\text{current}}(xt) = \left(\vec{D}_s \times x' \times y' \times Z'\right) + \vec{E}_{\text{best}}(x)$
13) x←x + 1
14) end while
15) return $\vec{E}_{best}$
16) end procedure

4.4.3 *Algorithm: Proposed SOA-CS*

1) Initialize the number of nests and fix the abandoned probability
2) Initialize population by fixing a lower bound and an upper bound.
3) Deduce the present best solution
4) Begin iterations:
5) Define the movement feature M in accordance to Eq. (6)
6) Deduce new solutions and retain the current best
   a. By applying Eqs. (2) and (3) deduce the Levy steps
7) Update the position by applying Eq. (7)
8) Deduce the Lévy coefficients as defined in Eq. (15)
9) Track the spiral movement in the x y z plane in accordance to Eq. (16)
10) Deduce the current best by comparing with the one obtained in Step 3
11) Abandon the worst solution with a p
12) Repeat from Step 4, till the required result is ascertained.

4.4.4 Flowchart: Proposed SOA-CS



5 Result and Discussion

5.1 Experimental Set-up

This section describes the simulated experiments that were conducted to ascertain the performance of the proposed SOA-CS algorithm. The Cloudsim 3.0 toolkit which has its base implemented in JAVA had been used for simulation purposes. These simulations had been done and authorized on a PC that is equipped with Intel(R) Core (TM) i5-457, 4 CPU @ 2.9 GHz, a RAM of 8 GB and 64-bit Windows OS. The simulation results that had yielded minimum values for cost and time had been taken up for discussion here. The efficiency of the proposed SOA-CS algorithm had been validated by comparing its results with Particle swarm optimization (PSO), Ant colony optimization (ACO) and improved differential evolution algorithm (IDEA) based on makespan and cost parameters. The parameter setup for the simulation is shown below in Tab. 3. Experiments had been conducted using 100, 200 and 300 VMs, with each distribution taking up 100–1000 tasks Assudani et al. [17], Abdullahi et al. [18].

Table 3: Parameter setup

Parameter	Value
Number of data centre	5
Number of VMs (M)	50
Type of VM policy	Time shared
Number of virtual CPUs	[1–5]
Size of VM RAM	[512–2048] MB
Capacity of virtual CPU	[500–2500] MIPS
Total number of tasks (N)	[100–1000]
VMM/Hypervisor	Xen

(Continued)

Table 3: Continued

Parameter	Value
VM operating system	Linux
Bandwidth	[250–1500]

5.2 Evaluation of Makespan

This section presents the comparative results for makespan parameter that were obtained using the proposed SOA-CS, PSO, ACO and IDEA algorithms. Here tasks were varied from 100 to 1000 numbers and the same were assigned to 100, 200 and 300 numbers of virtual machines. Work was accounted for 25 iterations. The makespan values obtained for 100, 200 and 300 VMs using the proposed SOA-CS approach and PSO, ACO, IDEA algorithms are represented in Figs. 2–4.

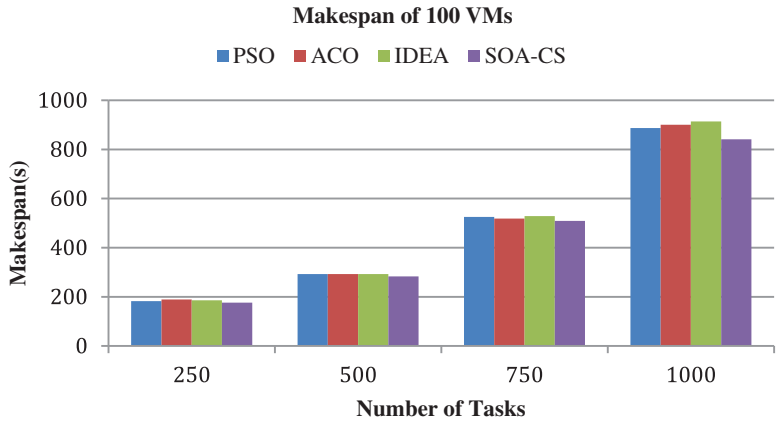


Figure 2: Makespan evaluation using 100 VMs

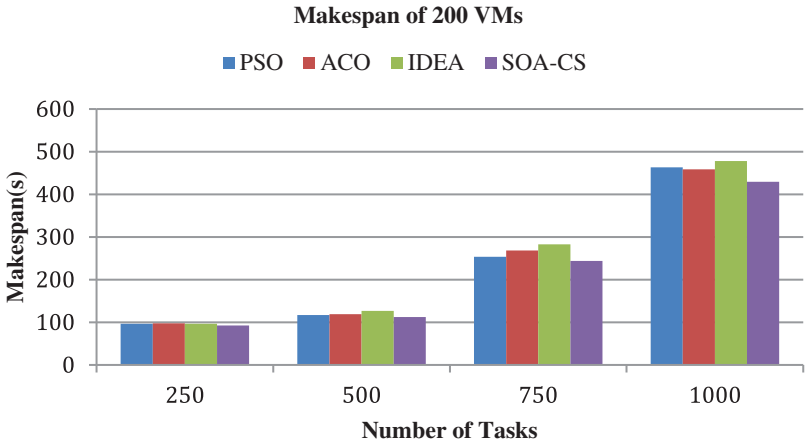


Figure 3: Makespan evaluation using 200 VMs

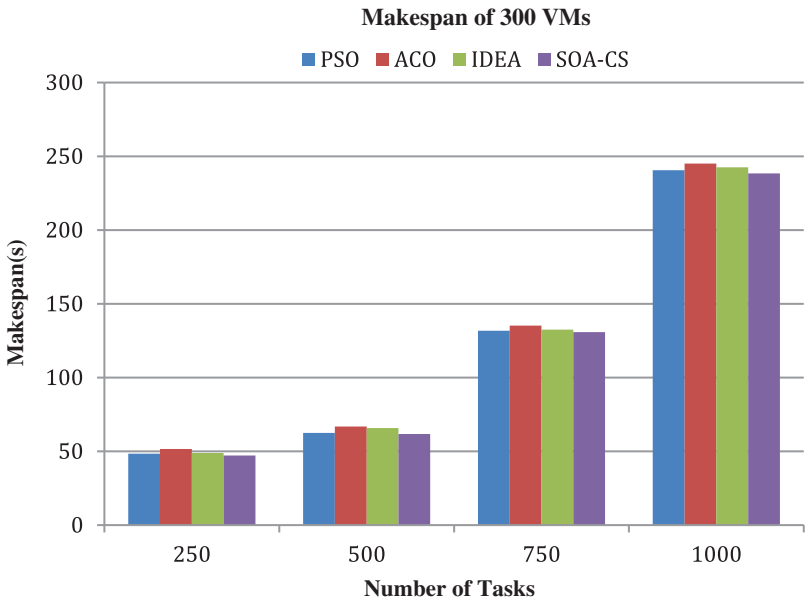


Figure 4: Makespan evaluation using 300 VMs

It could be inferred from Fig. 2 (using 100 VMs), that for 250 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 176.18(s), 182.4(s), 189.15(s) and 185.96(s) respectively. Similarly for 500 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 282.98 (s), 292.53(s), 292.64(s) and 292.78(s) respectively. For 750 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 509.02(s), 525.18(s), 518.66(s) and 528.74(s) respectively. For 1000 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 840.98(s), 886.96(s), 900.339(s) and 913.733(s) respectively. The proposed SOA-CS technique had produced an overall improvement of 4.30%, 5.06%, and 6.19% for makespan when 100 vms are considered compared with PSO, ACO, IDEA algorithms respectively.

It could be inferred from Fig. 3 (using 200 VMs), that for 250 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 92.68(s), 96.68(s), 97.5(s) and 96.54(s) respectively. Similarly for 500 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 112.19(s), 117.07(s), 119.08(s) and 126.82(s) respectively. For 750 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 243.9(s), 253.65(s), 268.29(s) and 282.92(s) respectively. For 1000 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 429.26(s), 463.41(s), 458.53(s) and 478.04(s) respectively. The proposed SOA-CS technique had produced an overall improvement of 6.01%, 7.44%, and 12.1% for makespan when 200 vms are considered compared with PSO, ACO, IDEA algorithms respectively.

It could be inferred from Fig. 4 (using 300 VMs), that for 250 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 47.2(s), 48.4(s), 51.6(s) and 48.9(s) respectively. Similarly for 500 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 61.8(s), 62.5(s), 66.8(s) and 65.8(s) respectively. For 750 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 130.8(s), 131.8(s), 135.2(s) and 132.5(s) respectively. For 1000 tasks, the makespan values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 238.4(s), 240.6(s), 245.1(s) and 242.5(s) respectively. The proposed SOA-CS technique had produced an

overall improvement of 1.06%, 4.2%, and 2.4% for makespan when 300 vms are considered compared with PSO, ACO, IDEA algorithms respectively.

5.3 Evaluation of Cost

This section presents the comparative results for cost parameter that were obtained using the proposed SOA-CS, PSO, ACO and IDEA algorithms. Here too, tasks were varied from 100 to 1000 numbers and were assigned to 100, 200 and 300 numbers of virtual machines, Natesan et al. [19,20]. Work was accounted for 25 iterations. The cost values obtained for 100, 200 and 300 VMs using the proposed SOA-CS approach and PSO, ACO, IDEA algorithms are represented in Figs. 5–7.

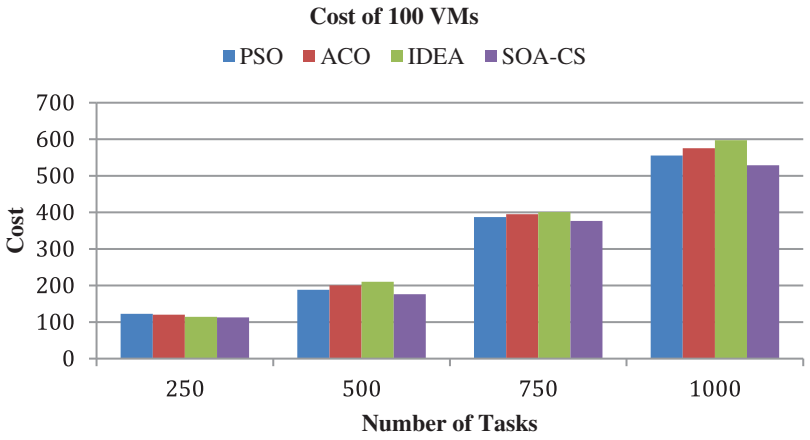


Figure 5: Cost evaluation using 100 VMs

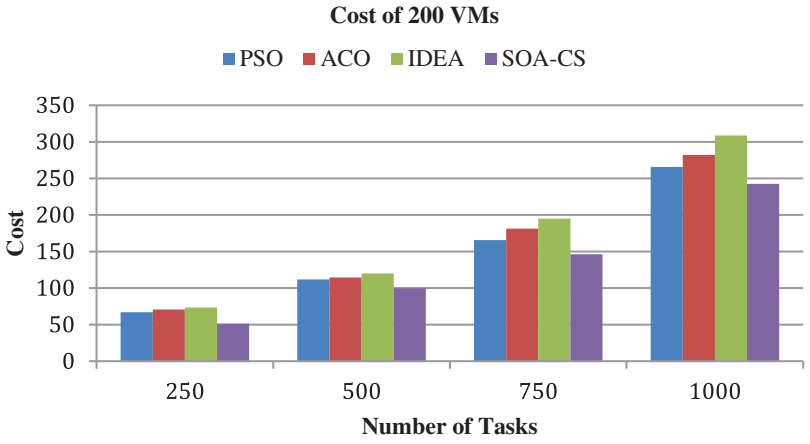


Figure 6: Cost evaluation using 200 VMs

It could be inferred from Fig. 5 (using 100 VMs), that for 250 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 112.7, 122.45, 120.36 and 114.23 respectively. Similarly for 500 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 176.07, 188.37, 200.37 and 210.35 respectively. For 750 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 376.69, 387.1, 394.97 and 400.93 respectively. For 1000 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 529.05, 555.43, 575.47 and

597.53 respectively. The proposed SOA-CS technique had produced an overall improvement to the extent of 4.9%, 8.09% and 10.7% for cost when 100 vms are considered when compared with PSO, ACO, IDEA algorithms respectively.

It could be inferred from Fig. 6 (using 200 VMs), that for 250 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 51.25, 66.97, 70.61 and 73.33 respectively. Similarly for 500 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 99.7, 111.75, 114.47 and 119.94 respectively. For 750 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 146.31, 165.72, 181.29 and 195.03 respectively. For 1000 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 242.53, 265.6, 282.113 and 308.7 respectively. The proposed SOA-CS technique had produced an overall improvement to the extent of 13.01%, 20.13% and 29.12% for cost when 200 vms are considered when compared with PSO, ACO, IDEA algorithms respectively.

It could be inferred from Fig. 7 (using 300 VMs), that for 250 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 34.2, 35.5, 37.7 and 36.7 respectively. Similarly for 500 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 57.2, 58.6, 59.2 and 59.1 respectively. For 750 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 84.5, 85.7, 87.1 and 86.5 respectively. For 1000 tasks, the cost values obtained for the proposed SOA-CS, PSO, ACO and IDEA are 134.2, 135.7 138.3 and 136.4 respectively. The proposed SOA-CS technique had produced an overall improvement to the extent of 1.74%, 3.93% and 2.77% for cost when 300 vms are considered when compared with PSO, ACO, IDEA algorithms respectively.

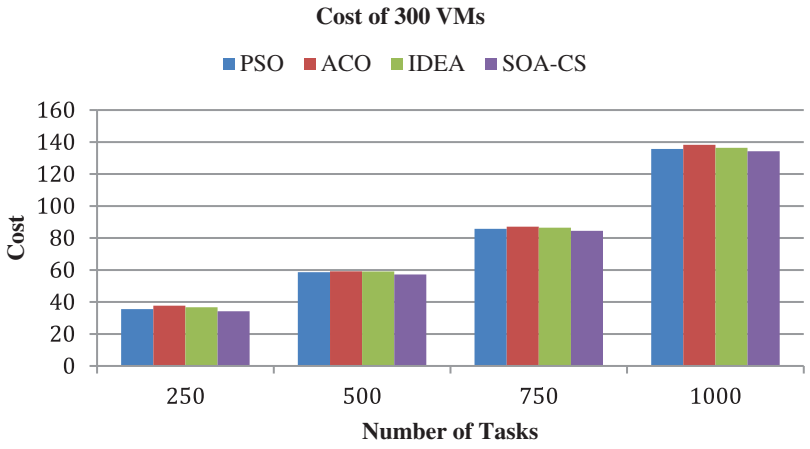


Figure 7: Cost evaluation using 200 VMs

6 Conclusion

The proposed SOA-CS algorithm has been designed by hybridizing the Seagull Optimization algorithm and the Cuckoo Search algorithms. The combined advantages of both the algorithms have been included in the hybridized design such that the convergence speed gets increased considerably. From the results obtained by comparing the proposed SOA-CS algorithm with the PSO, ACO and IDEA algorithms, it could be seen that the proposed SOA-CS produces an improvement of 1.06%, 4.2%, and 2.4% for makespan and had reduced the overall cost to the extent of 1.74%, 3.93% and 2.77% when compared with PSO, ACO, IDEA algorithms respectively when 300 vms are considered. Similarly, improvements for makespan and cost too are achieved when 100 and 200 vms had been

considered. In future, additional QoS parameters like reliability, load imbalance, fault tolerance and security could be considered to assess the performance efficiency of the proposed SOA-CS algorithm in real cloud environment.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 275–295, 2015.

[2] X. Zuo, G. Zhang and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, 2013.

[3] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[4] M. Nanjappan, G. Natesan and P. Krishnadoss, "An adaptive neuro-fuzzy inference system and black widow optimization approach for optimal resource utilization and task scheduling in a cloud environment," *Wireless Personal Communications*, vol. 121, no. 3, pp. 1891–1916, 2021.

[5] 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, pp. 1–20, 2021.

[6] P. Krishnadoss, N. Pradeep, J. Ali and M. Nanjappan, "CCSA: Hybrid cuckoo crow search algorithm for task scheduling in cloud computing," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 241–250, 2021.

[7] D. Alsadie, "A metaheuristic framework for dynamic virtual machine allocation with optimized task scheduling in cloud data centers," *IEEE Access*, vol. 9, pp. 74218–74233, 2021.

[8] Y. Gao, S. Zhang and J. Zhou, "A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS cloud," *IEEE Access*, vol. 7, pp. 125783–125795, 2019.

[9] P. Albert and M. Nanjappan, "WHOA: Hybrid based task scheduling in cloud computing environment," *Wireless Personal Communications*, vol. 121, no. 3, pp. 2327–2345, 2021.

[10] S. Mangalampalli, S. K. Swain and V. K. Mangalampalli, "Multi objective task scheduling in cloud computing using cat swarm optimization algorithm," *Arabian Journal for Science and Engineering*, vol. 47, no. 2, pp. 1821–1830, 2022.

[11] L. Zuo, L. Shu, S. Dong, C. Zhu and T. Hara, "A Multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing," *IEEE Access*, vol. 3, pp. 2687–2699, 2015.

[12] T. S. Somasundaram and K. Govindarajan, "CLOUDRB: A framework for scheduling and managing high-performance computing (HPC) applications in science cloud," *Future Generation Computer Systems*, vol. 34, pp. 47–65, 2014.

[13] G. Yao, Y. Ding, Y. Jin and K. Hao, "Endocrine-based coevolutionary multi-swarm for multi-objective workflow scheduling in a cloud system," *Soft Computing*, vol. 21, no. 15, pp. 4309–4322, 2017.

[14] K. Sreenu and M. Sreelatha, "W-scheduler: Whale optimization for task scheduling in cloud computing," *Cluster Computing*, vol. 22, no. 1, pp. 1087–1098, 2019.

[15] X. S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.

[16] G. Das and R. Panda, "Seagull-cuckoo search algorithm for function optimization," in *6th Int. Conf. for Convergence in Technology*, pp. 1–6, 2021.

[17] P. J. Assudani and P. Balakrishnan, "An efficient approach for load balancing of VMs in cloud environment," *Applied Nanoscience*, pp. 1–14, 2021.
[18] M. Abdullahi and M. A. Ngadi, "Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment," *PLoS One*, vol. 11, no. 6, pp. e0158229, 2016.
[19] H. He, G. Xu, S. Pang and Z. Zhao, "AMTS: Adaptive multi-objective task scheduling strategy in cloud computing," *China Communications*, vol. 13, no. 4, pp. 162–171, 2016.
[20] 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, 2019.