Computers, Materials & Continua
DOI:10.32604/cmc.2022.021793
Article

Tech Science Press

Hybrid Cuckoo Search Algorithm for Scheduling in Cloud Computing

Manoj Kumar* and Suman

CSE Department, DCR University of Science and Technology, Murthal, 131039, Sonepat, India
*Corresponding Author: Manoj Kumar. Email: manojsarasyiya@gmail.com
Received: 15 July 2021; Accepted: 27 August 2021

Abstract: Cloud computing has gained widespread popularity over the last decade. Scheduling problem in cloud computing is prejudiced due to enormous demands of cloud users. Meta-heuristic techniques in cloud computing have exhibited high performance in comparison to traditional scheduling algorithms. This paper presents a novel hybrid Nesterov Accelerated Gradient-based Cuckoo Search Algorithm (NAGCSA) to address the scheduling issue in cloud computing. Nesterov Accelerated Gradient can address trapping at local minima in CSA by updating the position using future approximation. The local search in the proposed algorithm is performed by using Nesterov Accelerated Gradient, while the global search is performed by using levy flights. The amalgamation of NAG and CSA helps in cost reduction and time-saving for users. The simulation has been carried out on the CloudSim tool on three different real datasets; NASA, HPC2N, and SDSC. The results of the proposed hybrid algorithm have been compared with state-of-art scheduling algorithms (GA, PSO, and CSA), and statistical significance is carried on mean, standard deviation, and best for each algorithm. It has been established that the proposed algorithm minimizes the execution cost and makespan, hence enhancing the quality of service for users.

Keywords: Cloud computing; scheduling; quality of service; cuckoo search; cost; makespan

1 Introduction

Cloud computing has become a fundamental part of the computing world nowadays. The expansion of cloud computing technology act as the backbone of newly developed technology like Fog Computing [1], Internet of Things (IoT) [2], Cloud of Things [3] etc. Cloud computing provides services to its users through the internet without knowledge of hardware and other infrastructures. Cloud uses on-demand policies, i.e., the services are available to its users as per their demand, and user has to pay to the service provider based on usage of cloud services [4]. However, there are methods of resource provisioning like leased and reserved, which are used by big organizations [5]. The additional benefits of cloud computing are scalability, availability, reliability, flexibility, and sustainability [6,7].

The key technology behind the success of cloud computing is virtualization. Virtualization helps cloud providers to deliver any kind of computing phenomenon by assembling the required hardware and software together on a single platform and minimizing the complexity and burden of buying new hardware and software. Some of the virtualization vendors are Xen [8], Virtual Box [9], VMWare Server [10]. Mostly offered public clouds are owned by Google Cloud [11], Amazon EC2 [12], Microsoft Azure [13].

Resource scheduling is a building block in cloud computing that decides how resources are distributed over time and made available to its users in the best possible way. It helps in providing the best quality of service to its users and the best utilization of resources to increase profits. The fluctuating demands of users are unpredictable concerning the time domain, so it is very difficult to satisfy resource requirements in dynamic nature. Scheduling in cloud computing is done at two levels: users to virtual machines (VM) and VM to the physical host. The first level scheduling is responsible for maintaining the quality of service in terms of makespan, execution cost, and response time while the latter decides load balancing, migrations, energy and resource utilization [14].

As the number of users and virtual machines keeps on changing continuously in cloud computing, so resource scheduling in the cloud system is an NP-Hard problem. Suppose there exit N users and M virtual machines for a broker to satisfy the user's requirements. The number of schedules for N users for M machines is N^2M which increases exponentially in the cloud environment with an increase in the number of machines and number of users. Therefore, resource scheduling is NP-Hard in nature and requires efficient algorithms that satisfy both users and cloud service providers. The fitness function of any scheduling algorithm is focused on cloud service providers and user requirements. The fitness function considers the availability, cost, makespan, energy, throughput, reliability, fault tolerance, etc., for optimal scheduling. An optimization problem can be stated as a fitness function f mapping the candidate solution to fitness measure $= \mathbf{R^n} \rightarrow \mathbf{R}$. The optimization solution $c \in R^n$ achieves the best optimal solution from all feasible solutions. Therefore, it can satisfy the minimization by Eq. (1) [15].

$$f(x) < f(y), \quad \forall (x, y) \in R^n \tag{1}$$

Hybrid algorithms always try to conquer all factors in solving any optimization problem. The benefits of combining two or more algorithms are that they always find the best solutions by addressing the limitations of the single algorithm in terms of speed and accuracy. Hybrid algorithms consist of some meta-heuristic techniques, while other methods can achieve optimal scheduling.

Cuckoo Search (CS) algorithm is a population-based meta-heuristic technique that has been widely used by the researcher in various domains of engineering and technology like scheduling, image processing, structural design, speech recognition etc. [16]. The searching behaviour of CS does not guarantee finding an optimal solution for specific problems. CS may trap into local minima and may lead to premature convergence. To solve this problem in CS, the Nesterov Accelerated Gradient technique is used in local search to improve the convergence rate and identify more feasible solutions to schedule activities on virtual machines with the lowest execution cost and time, hence improving the quality of service (QoS) for customers. Therefore, this paper proposes a novel hybrid Nesterov Accelerated Gradient-based Cuckoo Search (NAGCSA) algorithm to enhance QoS in terms of makespan and execution cost.

The vital contributions of this paper are as follows:

- Frame the makespan and execution cost using mathematical models for scheduling as the fitness functions
- Hybridize the Nesterov Accelerated Gradient with Cuckoo Search for scheduling in IaaS cloud
- Design of hybrid NAGCS algorithm to address the proposed scheduling model.
- Proposed hybrid NAGCS algorithm is implemented in the CloudSim simulation tool.
- Comparability of proposed hybrid NAGCS algorithm with CS algorithm and meta-heuristic techniques like GA and PSO in terms of makespan and execution cost.

The remaining sections of this paper are organized as follows: Section 2 review some existing techniques that are used in scheduling for IaaS cloud computing. Problem is formulated in Section 3. Section 4 presents the mathematical models of performance metrics that is used in this research work. Section 5 presents the explanation for CS search, Nesterov Accelerated Gradient and hybrid NAGCS algorithm. Section 6 presents the simulations setups for scheduling. Results and discussions are furnished in Section 7, and finally, we conclude the paper in last with some valuable suggestions.

2 Related Work

There is a vast literature on scheduling techniques in cloud computing. Researchers and scientists have applied many computational techniques to perform scheduling in cloud computing considering various constraints like budget, deadlines. The various review articles published in the literature have shown a broad application of heuristic and meta-heuristics techniques to solve various challenges in cloud computing [17]. Some past studies like [18] show various requirements of QoS based on particle swarm optimization. Computational intelligence has been divided into two parts heuristic, and meta-heuristic and how they were applied in cloud computing is very well explained in studies [19].

Genetic Algorithm (GA) has been widely used in cloud computing for performing scheduling, load balancing, and VM allocation. Its modified version and hybrid form with other techniques were also used for efficient scheduling in cloud computing. Author in [20] proposed multi QoS GA-ACO to schedule tasks on VMs to minimize the execution time and improves resource utilization. Alla et al. [21] proposed dynamic dispatch queue-based task scheduling strategies optimized using fuzzy logic and particle swarm optimization (PSO) and simulated annealing with PSO to minimize waiting time, makespan and cost. The algorithms were tested on both synthetic and real data systems while disabling the migration systems. The proposed work was extended to add energy parameters to minimize makespan while deadline as a constraint in [22]. The proposed algorithm improved QoS, minimize energy and improve resource utilization significantly.

Abari et al. [23] enhanced GA to perform static scheduling for a processor in a heterogeneous cloud computing environment by replacing random population with some initial population with relatively optimized solutions to lower repetition. Keshanchi et al. [24] modified GA with Linear Temporal Logic (LTL) formulas to schedule workflow in the cloud. The modified GA helps in prioritizing the tasks and determined the best available resource for execution efficiently. Zhou et al. [25] also modified the GA with a greedy strategy for scheduling the tasks in the cloud computing environment. The proposed method finds the solution very early in few iterations and helps minimise the makespan, response time, and maintain good QoS from the user's perspective.

Strumberger et al. [26] developed a scheduling algorithm with hybridization of monarch butterfly algorithm with ABC method to minimize the makespan. The algorithm was tested first on standard mathematical benchmark functions and then applied to scheduling problems in cloud sim simulator. Both synthetic and real data set were used to prove the performance of proposed algorithm; however, how priority of task is taken is not explained in work. Author also applied [27] whale optimization algorithm for scheduling of task by modifying the exploration stages of whale optimization with ABC and firefly to remove the weakness of Whale optimization algorithm. The algorithm achieves good QoS in terms of makespan and resource utilization.

Al-Olimat et al. [28] proposed the hybrid PSO-SA based scheduling technique to schedule the tasks in cloud computing environment. The SA helps in adjusting the weights of particles and moved them toward good solution. The proposed method minimizes the makespan however some more parameters need to be tested to find the performance of algorithm in multi-objective nature. Alsaih et al. [6] developed a dynamic job scheduling model to map the jobs with resource based on available resources characteristics and also re-schedule the jobs that get allocated to VM. Proposed method minimizes the makespan, improved CPU utilization and bandwidth utilization. Beegom et al. [29] developed discrete-integer based PSO algorithm to schedule tasks in cloud computing and applied it on single optimization and multiple optimization problem. The algorithm efficiently manages the degree of imbalance, minimize cost and makespan. Attiya et al. [30] designed a hybrid algorithm by combining the Harris Hawks Optimization and Simulated Annealing (SA) technique to minimize the makespan in cloud. The algorithm improves response time and converges early to give good solutions as search spaces increases.

Madni et al. [31] proposed multi-objective cuckoo search based scheduling to find the non-dominant solution in cloud computing to minimize execution cost, makespan and improves resource utilization. The simulations were carried out in CloudSim simulator, and real data set were used to evaluate the performance of algorithm with other meta-heuristic techniques. Author also extended the cuckoo search algorithm with gradient descent to increase the performance in [32]. Gradient descent was used to perform the local search, and levy flight in the cuckoo search was used to perform global search. The proposed hybrid algorithm was tested on both synthetic datasets and real datasets.

Kaur et al. in [33] proposed the hybrid GA-ACO based scheduling mechanism to minimize the execution time and makespan. The proposed algorithm was tested on a synthetic dataset inside Matlab and compare with GA and ACO; however, its performance can be compared with more meta-heuristic techniques.

The analysis of the reviewed articles established that the majority of studies includes meta-heuristic techniques like GA, PSO, ABC, and ACO and their hybridization with some heuristic techniques. These algorithms get trapped in local minima while performing scheduling in cloud computing. The performance of these algorithms is also not compared with other meta-heuristic techniques. This article presents a technique to avoid local optima problem using NAG and also performs better at global minima.

3 Problem Formulation

Resource scheduling in cloud computing is NP-Hard in nature. It is defined as mapping the user's task to the best available virtual machines to fulfill the user's demand. Suppose a cloud datacenter consists of p physical hosts, represented by set $P = \left\{P_1, P_2, P_3 \ldots\ldots\ldots\ldots.. P_p\right\}$. Each host Pi can have two or more virtual machines running on it. Let $VM_j = \{VM_1, VM_2, VM_3 \ldots\ldots\ldots\ldots VM_m\}$ represents the set of virtual machines. Each VM_j is a set of computing capabilities like CPU, memory, bandwidth, storage, and price. There are heterogeneous virtual machines, and their CPU capabilities (measured in millions of instructions per second) are used to find the expected execution time of tasks/cloudlets submitted by users. Suppose there is a set of activity/task/cloudlet $A_i = \{A_1, A_2, A_3 \ldots\ldots\ldots\ldots A_n\}$ submitted by users. Each activity A_i is a set of id, length, start time, file size. The processing requirement of an activity is referred as length of activity and is measured in millions of instructions. The expected execution time EET_{ij} of i^{th} activity on j^{th} virtual machine is calculated using Eq. (2).

$$EET_{ij} = \frac{Activity.Length_i}{VM.Power_j} \tag{2}$$

where $Activity.Length_i$ is the length of activity i and $VM.CPU_j$ is computing power of j^{th} VM in terms of MIPS. Therefore, EET of all the activities can be represented by Eq. (3).

$$\begin{bmatrix} A_1VM_1 & A_1VM_2 & A_1VM_3 & \ldots\ldots & A_1VM_m \\ A_2VM_1 & A_2VM_2 & A_2VM_3 & \ldots\ldots & A_2VM_m \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots. \\ A_nVM_1 & A_nVM_2 & A_nVM_3 & \ldots\ldots & A_nVM_m \end{bmatrix} \tag{3}$$

Suppose there is set of Activity cost $C_i = (C_1, C_2, C_3, \ldots\ldots\ldots\ldots C_n)$ that is made from cloud users as their demand per their demands. The broker is required to map all these activities to the virtual resources $VM_j = (VM_1, VM_2, VM_3 \ldots\ldots\ldots\ldots VM_m)$ with minimum cost and execution time. The expected execution cost of all the activities is represented using the EEC matrix in Eq. (4).

Selecting the best resource for cloud user with minimum cost in less time is very complex problem. To understand the complexity of the problem, a simple example is illustrated in Tab. 1.

Table 1: List of abbreviations

Symbol	Description
NAGCSA	Nesterov accelerated gradient cuckoo seach algorithm
VM	Virtual machines
QoS	Quality of service
GA	Genetic algorithm
PSO	Particle swarm optimization
EET	Expected execution time
EEC	Expected execution cost
PIR	Performance improvement rate

$$\begin{bmatrix} C_1VM_1 & C_1VM_2 & C_1VM_3 & \ldots\ldots & C_1VM_m \\ C_2VM_1 & C_2VM_2 & C_2VM_3 & \ldots\ldots & C_2VM_m \\ \ldots & \ldots & \ldots & \ldots\ldots & \ldots. \\ C_1VM_1 & C_nVM_2 & C_nVM_3 & \ldots\ldots & C_nVM_m \end{bmatrix} \tag{4}$$

Suppose there are two virtual machines available (VM1 and VM2) in the cloud datacenter and three users (A1–A3) requested for resource. The cloud broker can allocate any two users without waiting, but one user has to wait for some time. Considering this there are two choices available to broker, either any two users use machine 1 and one user use machine 2 or any two-user use machine

2 and only one user use machine 1. So, there are multiple decisions available to cloud broker and is responsible to give good quality of service to his users. The multiple decisions are illustrated in Tab. 2 with execution time and cost value. Similarly, more schedule S7 to S8 can also presented in which two users can use VM1 and remaining one will use VM2.

Table 2: 3 Jobs and 2 machines scheduling example

	VM2		VM1	Makespan, f(x)	Execution Cost, f(y)	Fitness
S1	A1	A2	A3	$47 + 47 + 74.6 + 95.33 = 263.93$	$3 + 6 + 4 = 13$	80.17
S2	A2	A1	A3	$74.6 + 74.6 + 47 + 95.33 = 291.53$	$6 + 3 + 4 = 13$	96.559
S3	A2	A3	A1	$74.6 + 74.6 + 47 + 78.33 = 284.7$	$6 + 3 + 4 = 13$	94.51
S4	A3	A2	A1	$57.2 + 57.2 + 74.6 + 78.33 = 267.33$	$6 + 4 + 3 = 13$	89.21
S5	A1	A3	A2	$47 + 47 + 57.2 + 124.3 = 308.2$	$3 + 3 + 6 = 12$	100.86
S6	A3	A1	A2	$57.2 + 57.2 + 47 + 124.33 = 285.73$	$3 + 3 + 6 = 12$	94.19

4 Mathematical Model for Scheduling

In this article, makespan time and execution cost are considered as the main objective for scheduling in cloud computing. Therefore, fitness value of hybrid NABCS can be calculated using equation Eqs. (5) and (6).

4.1 Makespan Model

Makespan is the maximum finishing time of all activities when they mapped on VMs. It is calculated using Eq. (5).

$$Makespan, f(x) = max \bigcup_{i=1}^{m} T_i, \quad \forall i \in N, \quad i = 1,\ 2,\ 3 \ldots .m \tag{5}$$

where T_i indicates the completion time of specific activity.

4.2 Execution Cost Model

Execution cost is defined as amount paid by cloud users to CSP for executing their activities on VMs as per Service Level Agreement (SLA). Eq. (6) is used for calculating execution cost for specific VM.

$$ExecutionCost, f(y) = \sum_{i=1}^{m} VM^i (C_i \times T_i), \quad \forall i \in N, \quad i = 1,\ 2,\ 3 \ldots ..m. \tag{6}$$

where C_i indicates the cost of VM^{ith} per unit time and T_i indicates time of utilization of that resource^i.

The two functions of makespan and execution cost can be combined together as weighted-product of two functions respectively. So, combining Eqs. (5), and (6) to into a single objective function:

$$FitnessFunction, F(X) = f(x)^{w} \times f(y)^{1-w} \tag{7}$$

where w is weight factor, that decides importance to makespan and execution cost, and $w \in [0, 1]$ .The Eq. (7) can also be rewritten as:

$$FitnessFunction, F(X) = e^{w.\ln(f(x))+(1-w).\ln(f(y))} \tag{8}$$

The Eq. (8) inherits the characteristics of weighted sum and weighted product performance metrics namely; makespan and execution cost. Combining both weighted sum and weighted product in calculation provides two advantages; firstly, it makes equation analytically tractable and logarithmic expectation is additive over time [34,35]. It helps to analyze makespan-cost trade-offs in much more efficient way to enhance QoS.

5 Methodology

In this section, the concepts of CS algorithm and NAG is presented. Then, hybridization of CS with NAG is presented to explain how NAG is helpful in local search.

5.1 Cuckoo Search

Cuckoo Search Algorithm [36] is a population-based meta-heuristic technique developed by Xin-She Yang and Suas Deb. The behaviour of various species of Cuckoo of laying their eggs in other's nests to maximize the hatching probability [36]. CS has significantly proved in global optimization problems for its fast convergence and speed. Various problems of engineering have been solved using CS, including scheduling. The original Cuckoo Search works on three principles: -

- Each Cuckoo lays one egg at a time and places it in a nest that has been selected at random.
- The highest-quality nest will be the carrier of the next generation or solution.
- The number of host nests is fixed, and a host can realize unfamiliar eggs with the probability (0 or 1). In this case, a host can abandon the nest to build a new nest or throw cuckoo egg.

There are three parameters used in CS algorithm.

- $P_\alpha \in [0, 1]$ probability worst nest to be abandoned.
- $\alpha > 0$ step size, depends on scale of the problem, mostly taken $\alpha > 1$.

$\rightarrow \lambda$ is random step length.

Levy flight is most commonly used in terms of step length with CS to update the solution as represented in Eq. (9)

$$x_i^{t+1} = x_i + \alpha \otimes Levy(\lambda) \tag{9}$$

where x_i^{t+1} is a new solution,

x_i is current solution,

$\alpha \bigotimes Levy(\lambda)$ is a transaction probability.

Assumptions of CS algorithm

(1) Cuckoo bird lays one egg at a time, which shows one solution.
(2) Nests having the best solution would be most delicate nests, i.e., nest itself represents the solution.
(3) A fixed number of the nests would be available; there are finite numbers of initial solutions, which will remain the same throughout the algorithm [36].

The behaviour of CS algorithm performs local and global search based on switching and discovery probability P_α. In most cases, the value of $P_\alpha = 0.25$ were taken for local search while a global search is explored more efficiently for $1 - P_\alpha$ time i.e., (0.75). The levy flight function helps perform the global search, and various studies have shown good convergence globally in many multi-discipline areas.

5.2 Nesterov Accelerated Gradient (NAG) Approach

Gradient Descent is commonly used optimization method for finding local minima of a function [37]. It has been widely used in machine learning to optimize any network. The gradient states that if a function **F(x)** is discrete and differentiable near any point **a,** the value of **F(x)** decreases if one move from a in negative of gradient of **F** at point **a,** $-\nabla\mathbf{F}\left(\mathbf{a}\right)$. So it can be stated that

$$b = a - \gamma \nabla F\left(a\right) \tag{10}$$

where a is current position, b is next position, γ is weight factor, and $\nabla F(a)$ is direction of Sleepest Ascent [38].

The concept of the Nesterov Accelerated Gradient [39] approach helps to update the function's position towards minimum with a look ahead. It helps in the correction of position if function moves away from the optimal solution in any particular iteration. The intuition behind accelerated gradient follows the rule-"look ahead before you leap" for a particular update. Now, applying this rule on Eq. (10),

$$b^i = a^i - update^i \tag{11}$$

For particular iteration i.

The $update^i$ can be derived from the momentum-based update rule and can be understood mathematically as follows:

(1) Without momentum

$$b^i = a^i - \gamma \nabla F(a^i) \tag{12}$$

(2) With momentum

$$update_i^a = \tau .update_{i-1}^a + \gamma \nabla F\left(a_i\right) \tag{13}$$

$$b = a - update_i^a \tag{14}$$

Now applying NAG rule on Eqs. (13) and (14), the look-ahead before you move can be calculated and updated as follows:

$$a_{lookahead} = a - \tau .update_{i-1}^a \tag{15}$$

$$update_i = \tau .update^{a}_{i-1} + \gamma \nabla \mathrm{a}_{\mathrm{lookahead}} \tag{16}$$

$$b^i = a^i - update_i \tag{17}$$

The updates come in two parts; a partial update and a final update. So, when a function overshoots and goes away from the optimal solution, NAG helps to take a U-turn and comes closely towards minima. The updating of a function from a particular point towards minima is based not only on current position but also on an approximation of future position. This helps in finding more accurate results and can be used with other meta-heuristic techniques with problem of trapping of a function at local minima.

5.3 Hybrid Nesterov Accelerated Gradient-Based Cuckoo Search Algorithm

Resource scheduling issues have been solved by various meta-heuristic techniques. Many scheduling algorithms have been proposed to optimize the scheduling issue in IaaS cloud computing, but every algorithm has limitations like speed or premature convergence. Most of the meta-heuristic techniques use the current position to update its position towards optimal solution. This hybrid algorithm uses the look-ahead position to move towards optimal solution even if it moves away from optimal solution. The proposed hybrid NAGCSA helps in local optimization due to NAG and global search maintains through breeding behaviour of cuckoo search algorithm.

A set of random integers is generated using an array that represents a solution for scheduling. This step finds dimension of solutions, given by number of *n* activities, lower bound and upper bound of the search space. Therefore, the process of generating $x_i \in X(i = 1, 2, 3 \ldots\ldots. N)$ is given by Eq. (18).

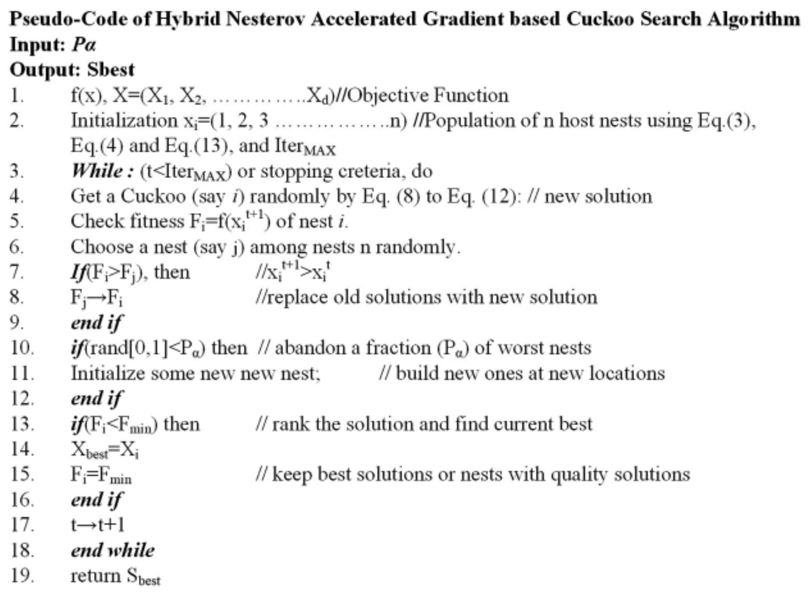
$$x_{ij} = Round\left(lb_{ij} + Rand\left(ub_{ij} - lb_{ij}\right)\right), \quad j = 1, 2, 3 \tag{18}$$

where each value of x_i has any integer value in the interval [1, m], where m is number of virtual machines, and *Round* coverts it to nearest value to whole number. To understand it more clearly, consider an example of twelve activities on five virtual machines. So, generated values for one solution using Eq. (13) are given in x_i = [4, 3, 2, 2, 1, 3, 1, 5, 2, 4, 3, 3]. The value “four” indicates the first job will be assigned to fourth machine. So, second, sixth, eleventh and twelfth jobs will be assigned to third machine. Similarly, interpretation of all other values can be done. The Eqs. (3) and (4) will calculate requires amount of execution time and cost initially. This x_i also represents cuckoo’s eggs in our algorithm.

After the initialization stage, fitness of all eggs is calculated, the best is selected. All cuckoos now fly towards the optimal solution with step-size variable. NAG is used for performing local search when $P_\alpha < 0.25$ as is done in Eqs. (8)–(12). The local search is carried out using abundant probability P_α. However, the global search is done using standard levy flight as in Eq. (7) for each iteration. The criteria to stop the iteration depends upon either fixing the number of iteration or achieving the required quality of solution. Fig. 1 represents proposed hybrid NAGCSA algorithm.

6 Simulation Setup

The simulation setup is described in this section. Simulation has been prepared on Apache NetBeans IDE 11.3, supported by CloudSim simulation tool 3.03 [40], which is a Java-based simulator. The software has been installed on the HP ProDesk desktop computer having 64-bit Windows 10, 16 GB RAM, 3.20 GHz CPU, with eight cores. The performance of the proposed hybrid NAGCSA is compared with genetic algorithm (GA), particle swarm optimization (PSO), and standard cuckoo search (CS) algorithms, respectively. Parameter setting of each meta-heuristic technique is presented in Tab. 3. The configuration settings of CloudSim are presented in Tab. 4.



Pseudo-Code of Hybrid Nesterov Accelerated Gradient based Cuckoo Search Algorithm
Input: *Pα*
Output: Sbest

```
1.  f(x), X=($X_1$, $X_2$, ……………..$X_d$)//Objective Function
2.  Initialization $x_i$=(1, 2, 3 ………………..n) //Population of n host nests using Eq.(3),
    Eq.(4) and Eq.(13), and $Iter_{MAX}$
3.  While : ($t<Iter_{MAX}$) or stopping creteria, do
4.  Get a Cuckoo (say i) randomly by Eq. (8) to Eq. (12): // new solution
5.  Check fitness $F_i=f(x_i^{t+1})$ of nest i.
6.  Choose a nest (say j) among nests n randomly.
7.  If($F_i>F_j$), then          //$x_i^{t+1}>x_i^t$
8.  $F_j\rightarrow F_i$                    //replace old solutions with new solution
9.  end if
10. if(rand[0,1]<$P_\alpha$) then  // abandon a fraction ($P_\alpha$) of worst nests
11. Initialize some new new nest;          // build new ones at new locations
12. end if
13. if($F_i<F_{min}$) then         // rank the solution and find current best
14. $X_{best}=X_i$
15. $F_i=F_{min}$                   // keep best solutions or nests with quality solutions
16. end if
17. $t\rightarrow t+1$
18. end while
19. return $S_{best}$
```

Figure 1: Hybrid nesterov accelerated gradient-based cuckoo search algorithm

Table 3: Parameter settings in meta-heuristics algorithms used in scheduling

Algorithms	Parameters	Values
GA	Population size	1000
	Max iteration	1000
	Cross-over rate	0.5
	Mutation rate	0.1
PSO	Particle size	100
	Self-recognition coefficients, c1, c2	2
	Uniform random number, R1	0,1
	Max iteration	1000
	Inertia weight	0.9–0.4
CS	Population size	50
	Abundant probability, *Pα*	0.25
	Step size, α	0.01, 1
	Learning rate, γ	0.9

The performance of the proposed hybrid NAGCSA is evaluated on real traces from HPC2N (High-Performance Computing Center North) [41], NASA Ames iPCS/860 [42], and SDSC (San Diego Supercomputer Center) [43]. These workload archives are presented by "Ake Sandgren, Bill Nitzberg and Victor Hazlewood", in the standard workload format (swf) recognized by the CloudSim tool. These workloads are primarily used to evaluate the performance of scheduling algorithms in cloud computing. HPC2N has 527,371 jobs, NASA has 14,794 jobs, and SDSC has 73,496 jobs.

Table 4: Parameters setting in CloudSim for cloud computing environment

Entities	Parameters	Values
Users	Number of users	50
	Number of broker	1
Cloudlet	Number of cloudlets/activities	200–2000
	Length	800,000
	File size	600 KB
VM	No. of VMs	50
	Type of policy	Space shared
	RAM	512–2048 MB
	Bandwidth	10000 Mbps
	MIPS	1000–2000
	Size	100–1000 GB
	VMM	Xen
	Operating system	Linux
	Number of CPUs	1 on each
Host	RAM	4096 MB
	Storage	100 TB
	Bandwidth	10000 Mbps

7 Results and Discussion

This section presents and discusses the computational outcomes of the simulation performed in CloudSim tool. The makespan time and execution cost are considered to evaluate the Quality of Service of the proposed algorithm.

Figs. 2–4 show the execution cost calculated of the hybrid NAGCSA algorithm with the help of NASA, HPC2N and SDSC workload. The simulation has been carried out on a varied number of cloudlets ranging from 500 to 2000. The other meta-heuristic algorithms used in resource scheduling (GA, PSO, and CSA) in cloud computing are used for comparison with the proposed hybrid algorithm. The pricing policy of VMs was taken from Google AppEngine [11], and the unit of cost is dollar ($). The execution cost calculated by the proposed hybrid algorithm is lower than the other algorithms. It can be seen that the proposed hybrid algorithm supports cloud users to save more money in a cloud environment. Execution cost increases while an increase in the number of cloudlets. The performance of the proposed hybrid algorithm is increasing while an increase in the number of cloudlets. The performance improvement rate of the proposed hybrid algorithm is shown in Tabs. 5–7.

The performance improvement rate (PIR) is calculated to evaluate the improvement for the x^{th} algorithm as compared to y^{th} algorithm [31,32,44]. It can be calculated using Eq. (19).

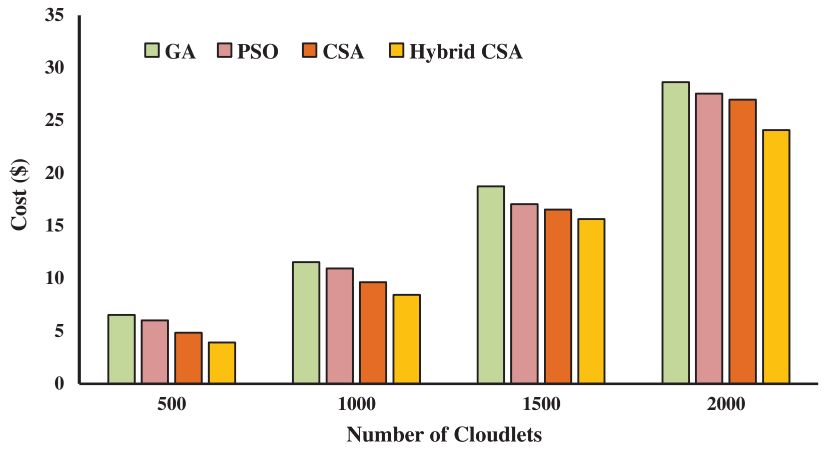


Figure 2: Execution cost on NASA dataset

$$PIR\,(\%) = \left(\frac{\left(\sum_{x^{th}} PM - \sum_{y^{th}} PM\right)}{\sum_{y^{th}} PM}\right) \times 100 \tag{19}$$

PIR (%) is calculated based on makespan on three different datasets shows that proposed hybrid Cuckoo Search Algorithm has a significant performance than GA, PSO, and CSA. Proposed hybrid algorithm minimizes execution cost by 35.59%, 32.22%, and 31.26% as compared to GA; 26.61%, 22.19%, and 24.05% as compared to PSO; and 13.83%, 11.40%, and 16.35% as compared to CSA on NASA, HPC2N, and SDSC workloads respectively, which is presented in Tab. 5–7.

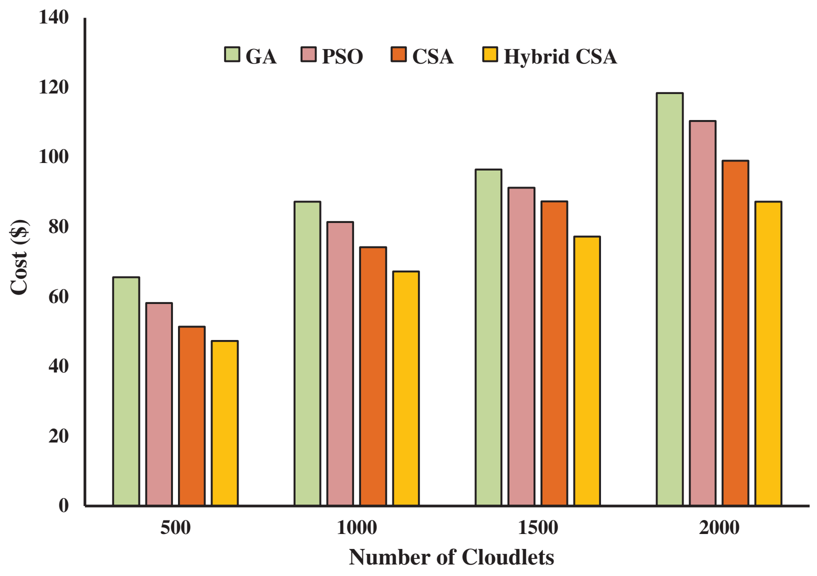


Figure 3: Execution cost on HPC2N dataset

Figs. 5–7 present the makespan calculated by a proposed hybrid algorithm for three different datasets in seconds. The other meta-heuristic algorithms, GA, PSO, and CSA, were also simulated with the same configuration to calculate the makespan. Figs. 5–7 show that our proposed hybrid NAGCSA algorithm minimizes the makespan and finishes the execution of all cloudlets earlier than GA, PSO, and CSA. The simulation has been carried out on a varied number of cloudlets from 500 to 2000, and a mean of 30 simulation results are presented in the form of bar charts. The performance of proposed hybrid algorithm saves more time. The performance improvement rate (%) in makespan is presented in Tabs. 8–10 for each dataset. Proposed hybrid algorithm reduces time and exhibits 19.82%, 13.54%, 7.21% improvement on NASA, 7.26%, 4.96%, and 3.20% improvement on HPC2N, 15.09%, 10.24%, and 8.54% improvement on SDSC over GA, PSO, and CSA respectively.

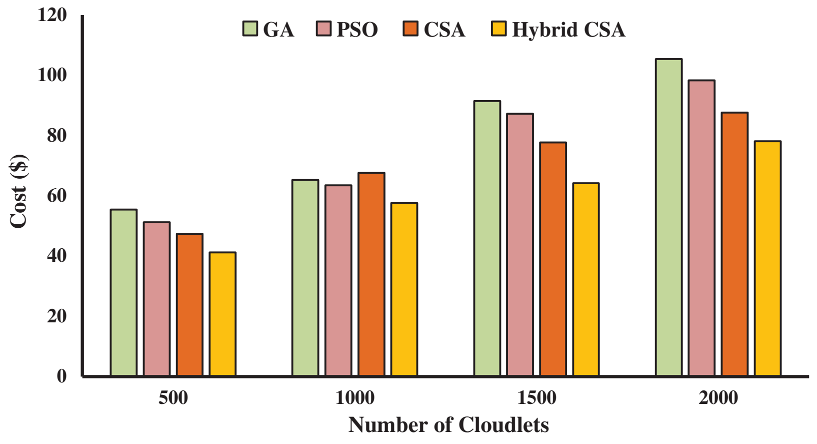


Figure 4: Execution cost on SDSC dataset

Table 5: PIR of execution cost on NASA workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	6.54	6.02	4.84	3.92	66.83	53.57	23.46
1000	11.56	10.95	9.65	8.45	36.80	29.58	14.20
1500	18.74	17.05	16.54	15.65	19.74	8.94	5.68
2000	28.65	27.54	26.97	24.08	18.97	14.36	12.00
Mean % improvement (a. NASA)					35.59	26.61	13.83

Table 6: PIR of execution cost on HPC2N workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	65.63	58.23	51.45	47.36	38.58	22.95	8.64
1000	87.25	81.45	74.23	67.23	29.78	21.15	10.41
500	96.45	91.25	87.36	77.25	24.85	18.12	13.09
2000	118.36	110.36	98.97	87.23	35.69	26.52	13.46
Mean % improvement (a. HPC2N)					32.22	22.19	11.40

Table 7: PIR of execution cost on SDSC workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	55.43	51.23	47.36	41.25	34.38	24.19	14.81
1000	65.25	63.45	67.65	57.63	13.22	10.10	17.39
1500	91.45	87.25	77.69	64.15	42.56	36.01	21.11
2000	105.36	98.36	87.58	78.12	34.87	25.91	12.11
Mean % improvement (c. SDSC)					31.26	24.05	16.35

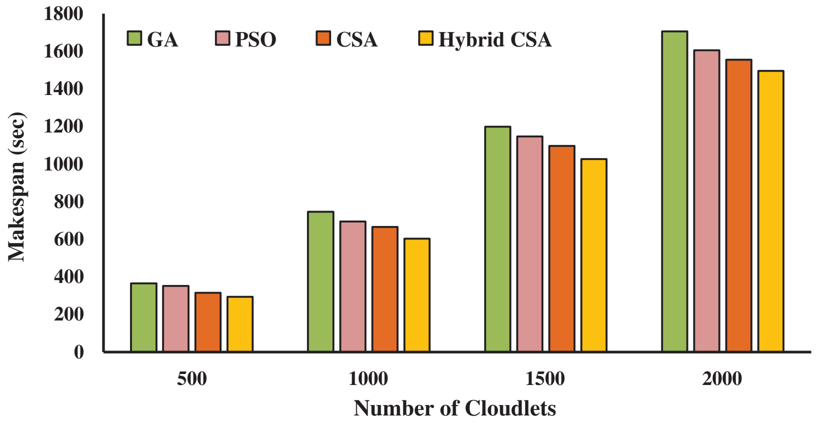


Figure 5: Makespan calculated on NASA dataset

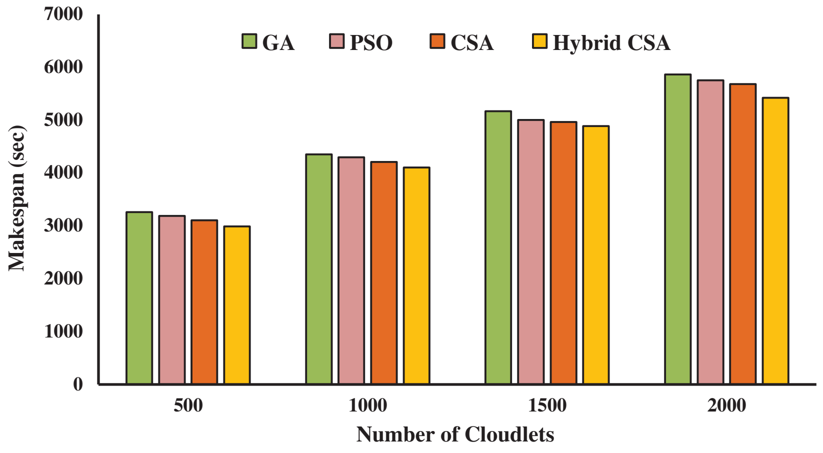


Figure 6: Makespan calculated on HPC2N dataset

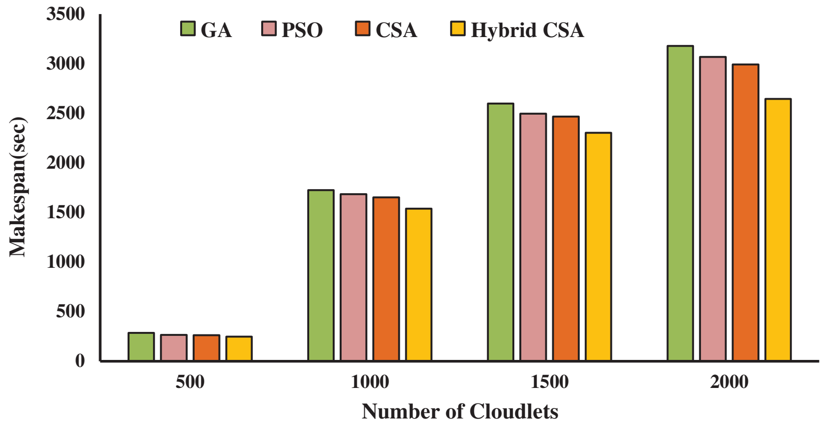


Figure 7: Makespan calculated on SDSC dataset

Table 8: PIR of makespan on NASA workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	364.41	350.98	315.14	292.65	24.52	19.93	7.68
1000	745.96	694.23	664.85	602.45	23.82	15.23	10.36
1500	1198.47	1145.45	1095.74	1025.45	16.87	11.70	6.85
2000	1705.45	1604.25	1554.23	1495.36	14.05	7.28	3.94
Mean % improvement (a. NASA)					19.82	13.54	7.21

Table 9: PIR of makespan on HPC2N workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	3258.49	3187.21	3106.45	2987.32	9.08	6.69	3.99
1000	4354.12	4296.41	4208.24	4105.36	6.06	4.65	2.51
1500	5168.54	5004.96	4965.87	4887.47	5.75	2.40	1.60
2000	5865.64	5754.25	5678.54	5423.74	8.15	6.09	4.70
Mean % improvement (b. HPC2N)					7.26	4.96	3.20

Table 10: PIR of makespan on SDSC workload

Jobs/ Cloudlets/ Activities	GA	PSO	CSA	Hybrid CSA	PIR over GA	PIR over PSO	PIR over CSA
500	285.65	265.47	263.54	247.95	15.20	7.07	6.29
1000	1725.45	1684.35	1654.23	1537.94	12.19	9.52	7.56
1500	2598.23	2496.23	2467.96	2304.54	12.74	8.32	7.09
2000	3180.36	3069.63	2994.45	2645.24	20.23	16.04	13.20
Mean % improvement (c. SDSC)					15.09	10.24	8.54

The statistical significance of each algorithm for execution cost and makespan after thirty simulation runs is presented in Tabs. 11 and 12. The comparison is made on mean, standard deviation, and best value is considered. Tabs. 11 and 12 signifies that our proposed hybrid NAGCSA is more significant in terms of execution cost and makespan.

Table 11: Statistical significance of execution cost after 30 runs

Statistical significance of execution cost after 30 runs										
Algorithm	Workload	NASA			HPC2N			SDSC		
	Number of cloudlet	$\overline{X}$	σ	Best	$\overline{X}$	σ	Best	$\overline{X}$	σ	Best
GA	500	6.54	0.24	6.31	65.63	0.39	63.54	55.43	2.13	53.69
	1000	11.56	0.96	10.89	87.25	0.86	85.98	65.25	1.86	62.36
	1500	18.74	0.54	18.71	96.45	0.61	95.36	91.45	1.64	89.36
	2000	28.65	1.41	27.36	118.36	1.54	117.25	105.36	1.47	103.45
PSO	500	6.02	0.36	5.91	58.23	0.47	57.52	51.23	0.59	49.69
	1000	10.95	1.54	9.64	81.45	1.87	80.36	74.45	1.97	72.45
	1500	17.05	1.26	16.78	91.25	1.68	90.36	87.25	2.95	84.36
	2000	27.54	1.58	26.45	110.36	1.71	106.69	98.36	2.76	94.36
CSA	500	4.84	0.14	4.70	51.45	0.25	50.36	47.36	0.39	46.36
	1000	9.65	0.95	8.90	74.23	0.87	72.25	67.65	0.56	65.25
	1500	16.54	0.71	16.02	87.36	0.89	84.36	77.69	0.72	76.36
	2000	26.97	1.25	25.52	98.97	1.64	96.36	87.58	1.64	85.23
Hybrid CSA	500	3.92	0.05	3.83	47.36	0.56	45.36	41.36	0.72	40.66
	1000	8.45	0.15	8.20	67.23	0.69	65.61	57.63	0.62	55.36
	1500	15.65	0.30	15.35	77.25	0.89	75.26	64.15	0.77	62.45
	2000	24.08	0.95	23.81	87.23	1.02	85.23	78.12	1.07	76.36

Table 12: Statistical significance of makespan after 30 runs

Statistical significance of makespan after 30 runs										
Algorithm	Workload	NASA			HPC2N			SDSC		
	Number of cloudlet	$\overline{X}$	σ	Best	$\overline{X}$	σ	Best	$\overline{X}$	σ	Best
GA	500	364.41	3.54	361.87	3258.49	9.25	3214.56	285.65	6.25	271.25
	1000	745.96	3.96	743.36	4354.12	10.69	4319.65	1725.45	7.25	1694.36
	1500	1198.47	4.58	1194.54	5168.54	12.87	5118.74	2598.23	7.21	2564.31
	2000	1705.45	4.98	1695.24	5865.64	14.36	5817.23	3180.36	8.54	3119.36
PSO	500	350.98	3.56	347.25	3187.21	8.69	3112.87	265.47	5.36	255.66
	1000	694.23	2.66	692.74	4296.41	11.65	4247.32	1684.35	6.32	1677.21
	1500	1145.45	3.98	1139.54	5004.96	13.58	4987.23	2496.23	6.35	2474.25
	2000	1604.25	2.49	1598.73	5754.25	14.36	5714.11	3069.63	7.88	3015.66
CSA	500	315.14	3.58	313.47	3106.45	8.69	3084.78	263.54	5.45	257.66
	1000	664.85	2.98	661.25	4208.24	9.35	4198.45	1654.23	5.98	1598.90
	1500	1095.74	2.57	1093.78	4965.87	10.54	4961.54	2467.96	6.36	2395.69
	2000	1554.23	2.60	1554.73	5678.54	11.26	5617.12	2994.45	6.87	2923.36
Hybrid CSA	500	292.65	1.25	291.78	2987.32	8.64	2917.23	247.95	4.36	245.66
	1000	602.45	1.95	600.57	4105.36	9.21	4084.23	1587.94	4.98	1577.21
	1500	1025.45	2.18	1023.66	4887.47	9.35	4824.23	2304.54	5.3	2285.66
	2000	1495.36	2.68	1493.54	5423.74	10.69	5396.36	2645.24	5.91	2654.21

8 Conclusions and Future Work

Scheduling in cloud computing is considered an NP-Hard problem to fulfil the demands of all users. It has been explored by most scientists and researchers around the world to increase the performance of scheduling algorithms. The motivation for this paper has explored techniques to study important parameters of quality of service, i.e., makespan and execution cost, for cloud users. This paper proposed a novel hybrid Nesterov Accelerated Gradient-based Cuckoo Search Algorithm to perform the scheduling in the IaaS cloud. The approximation of the future position of using NAG helps to reach the function towards local minima even if gets overshoots in any iteration. The simulation was performed in CloudSim on three different datasets; NASA, HPC2N, and SDSC with GA, PSO, and CSA scheduling algorithms. The results show that our proposed hybrid algorithm helps in saving time and money of users up to 19.20% and 35%, respectively. This work can be extended by adding more parameters like degree of imbalance, energy consumption, and throughput. The proposed hybrid algorithm can be used to resolve other issues in cloud computing like VM allocation and load balancing. More gradient-based optimization algorithms can be hybridized with CSA to evaluate its performance in other engineering optimizations problems.

Acknowledgement: Authors also acknowledge the Computer Science and Engineering Department of Deenbandhu Chhotu Ram University of Science and Technology for providing various facilities in the research lab.

Funding Statement: This research work is funded by University Grant Commission under National Fellowship Programme, Grant No. 2017-18/29146.

Conflicts of Interest: The authors declare that they have conflict of interest to report regarding this research work.

References

[1] H. Gupta, A. V. Dastjerdi, S. K. Ghosh and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.
[2] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Generation Computing System*, vol. 29, no. 7, pp. 1645–1660, 2013.
[3] M. Aazam, I. Khan, A. A. Alsaffar and E.-N. Huh, "Cloud of things: Integrating internet of things and cloud computing and the issues involved," in *Proc. IBCAST*, Islamabad, Pakistan, pp. 414–419, 2014.
[4] E. N. Al-Khanak, S. P. Lee, S. U. R. Khan, N. Behboodian, O. I. Khalaf *et al.,* "A heuristics-based cost model for scientific workflow scheduling in cloud," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3265–3282, 2021.
[5] M. Kumar, S. C. Sharma, A. Goel and S. P. Singh, "A comprehensive survey for scheduling techniques in cloud computing," *Journal of Network. Computer Applications*, vol. 143, no. 2, pp. 1–33, 2019.
[6] M. A. Alsaih, R. Latip, A. Abdullah, S. K. Subramaniam and K. Ali Alezabi, "Dynamic job scheduling strategy using jobs characteristics in cloud computing," *Symmetry*, vol. 12, no. 10, pp. 1638, 2020.
[7] P. Baldoss and G. Thangavel, "Optimal resource allocation and quality of service prediction in cloud," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 253–265, 2021.
[8] Xen, "Xen Project," (accessed Apr. 29, 2021). [Online]. Available: https://xenproject.org/.
[9] Oracle VM VirtualBox, (accessed May 07, 2021). [Online]. Available: https://www.virtualbox.org/.
[10] VMware India-delivering a digital foundation for businesses | IN, (accessed May 07, 2021). [Online]. Available: https://www.vmware.com/in.html.
[11] Google AppCloud, (accessed May 07, 2021). [Online]. Available: https://cloud.google.com/.
[12] Amazon EC2, (accessed May 07, 2021). [Online]. Available: https://aws.amazon.com/.
[13] Microsoft Azure: Cloud computing services, (accessed May 07, 2021). [Online]. Available: https://azure.microsoft.com/en-in/.
[14] M. Kumar, Suman and S. Singh, "A survey on virtual machine scheduling algorithms in cloud computing," *International Journal of Computer Sciences and Engineering*, vol. 6, no. 3, pp. 485–490, 2018.
[15] D. D. Zhu, P. M. Pardalos, W. Wu, C. Floudas and P. Pardalos, History of optimization. In: *Encyclopedia of Optimization*. Berlin: Springer, pp. 1538–1542, 2009.
[16] N. Dey, X. S. Yang and S. Fong, *Applications of Cuckoo Search Algorithm and its Variants*. Berlin: Springer, pp. 1–324, 2021.
[17] M. Kumar and Suman, "Meta-heuristics techniques in cloud computing: Applications and challenges," *Indian Journal of Computer Science and Engineering*, vol. 12, no. 2, pp. 385–395, 2021.
[18] M. Farid, R. Latip, M. Hussin and N. A. W. A. Hamid, "A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing," *Symmetry*, vol. 12, no. 4, pp. 551, 2020.
[19] A. R. Arunarani, D. Manjula and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Generation Computers System*, vol. 91, no. 4, pp. 407–415, 2019.

[20] Y. Dai, Y. Lou and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing," in *Proc. ICIHMSSC*, Hangzhou, China, pp. 428–431, 2015.

[21] H. B. Alla, S. B. Alla, A. Touhafi and A. Ezzati, "A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment," *Cluster. Computing*, vol. 21, no. 4, pp. 1797–1820, 2018.

[22] S. B. Alla, H. B. Alla, A. Touhafi and A. Ezzati, "An efficient energy-aware tasks scheduling with deadline-constrained in cloud computing," *Computers*, vol. 8, no. 2, pp. 46, 2019.

[23] M. Akbari, H. Rashidi and S. H. Alizadeh, "An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems," *Engineering Applications of Artificial Intelligence*, vol. 61, pp. 35–46, 2017.

[24] B. Keshanchi, A. Souri and N. J. Navimipour, "An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing," *Journal of Systems and Software*, vol. 124, no. 8, pp. 1–21, 2017.

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

[26] I. Strumberger, M. Tuba, N. Bacanin and E. Tuba, "Cloudlet scheduling by hybridized monarch butterfly optimization algorithm," *Journal of Sensor and Actuator Networks*, vol. 8, no. 3, pp. 1–39, 2019.

[27] I. Strumberger, N. Bacanin, M. Tuba and E. Tuba, "Resource scheduling in cloud computing based on a hybridized Whale optimization algorithm," *Applied Sciences*, vol. 9, no. 22, pp. 4893, 2019.

[28] H. S. Al-Olimat, M. Alam, R. Green and J. K. Lee, "Cloudlet scheduling with particle swarm optimization," in *Proc. ICCSNT*, Gwalior, India, pp. 991–995, 2015.

[29] A. S. A. Beegom and M. S. Rajasree, "Integer-PSO: A discrete PSO algorithm for task scheduling in cloud computing systems," *Evolutionary Intelligence*, vol. 12, no. 2, pp. 227–239, 2019.

[30] I. Attiya, M. A. Elaziz and S. Xiong, "Job scheduling in cloud computing using a modified Harris Hawks optimization and Simulated Annealing algorithm," *Computational Intelligence and Neuroscience*, vol. 2020, no. 16, pp. 1–17, 2020.

[31] S. H. H. Madni, M. S. A. Latiff, J. Ali and S. M. Abdulhamid, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds," *Arabian Journal of Science and. Engineering*, vol. 44, no. 4, pp. 3585–3602, 2019.

[32] S. H. H. Madni, M. S. Abd Latiff, S. M. Abdulhamid and J. Ali, "Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment," *Cluster Computing*, vol. 22, no. 1, pp. 301–334, 2019.

[33] M. Kaur and M. Agnihotri, "Performance evaluation of hybrid GA-ACO for task scheduling in cloud computing," in *Proc. ICCCI*, Greater Noida, India, pp. 168–172, 2016.

[34] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 2018.

[35] A. Gandhi, V. Gupta, M. H. Balter and M. A. Kozuch, "Optimality analysis of energy-performance trade-off for server farm management," *Performance Evaluation*, vol. 67, no. 11, pp. 1155–1171, 2010.

[36] X. S. Yang and S. Deb, "Cuckoo search via levy flights," in *Proc. WCNBIC*, Coimbatore, India, pp. 210–214, 2009.

[37] J. A. Snyman, Practical mathematical optimization. In: *An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. Berlin: Springer, 2005.

[38] R. Fletcher and M. J. D. Powell, "A rapidly convergent descent method for minimization," *The Computer Journal*, vol. 6, no. 2, pp. 163–168, 1963.

[39] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematics Program*, vol. 140, no. 1, pp. 125–161, 2013.

[40] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.
[41] Parallel workloads archive: HPC2N seth, (accessed May 04, 2021). [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/l_hpc2n/.
[42] Parallel workloads archive: NASA iPSC/860, (accessed May 04, 2021). [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/1_nasa_ipsc/.
[43] Parallel workloads archive: SDSC SP2, (accessed May 04, 2021). [Online]. Available: https://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_sp2/.
[44] M. Kumar and Suman, "Energy efficient scheduling in cloud computing using black widow optimization," *Journal of Physics: Conference Series*, vol. 1950, pp. 12063, 2021.