Received: February 19, 2021. Revised: May 5, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

CCSA: Hybrid Cuckoo Crow Search Algorithm for Task Scheduling in Cloud Computing

Pradeep Krishnadoss[1]* Gobalakrishnan Natesan[1] Javid Ali[1]
Manikandan Nanjappan[1] Parkavi Krishnamoorthy[2] Vijayakumar Kedalu Poornachary[2]

[1,2]Vellore Institute of Technology, Chennai, Tamilnadu, India
[1,1]St.Joseph's College of Engineering, Chennai, Tamilnadu, India
[1]St.Joseph's Institute of Technology, Chennai, Tamilnadu, India
*Corresponding author's Email: pradeepkrishnadoss@gmail.com

Abstract: Services are delivered promptly and efficiently managed over the Internet using cloud computing. Data and storage services are offered to users through cloud computing without regard to the actual physical location of the users. The cloud environment includes huge number of tasks and computing resources. Identification of appropriate virtual machine (VM) for allocation of resources to complete any submitted task is done through the task scheduling algorithms, which play a significant role in the cloud computing process. Task scheduling techniques improvise the makespan of the schedule and account for considerable reduction of cost expended. In this paper, an efficient hybridized scheduling algorithm that replicates the parasitic behaviour of the cuckoo and food gathering habit of the crow bird, named Cuckoo Crow Search Algorithm (CCSA) had been presented for improvising the task scheduling process. The crow bird always stares at its neighbours looking for a better food source than the one it currently possesses. At some circumstances the crow even goes a step further and steals its neighbour's food. The CCSA had been inspired from such characteristics of these birds and had been designed to be applied in the cloud environment for identifying a suitable VM for carrying out the task scheduling process. The proposed CCSA accounts for reduction in makespan and cost and its performance had been experimentally verified by comparing it with other existing algorithms like Multi objective - Ant Colony Optimization (MO-ACO), ACO, Min-Min and the results had been presented. The proposed CCSA technique had produced an improvement of 3.14%, 10.70%, and 21.60% for makespan and had reduced the overall cost to the tune of 4.56%, 11.19% and 19.35% when compared with MO-ACO, ACO, Min-Min algorithms respectively when used with 10 VMs. The detailed description of results obtained had been described in result section.

Keywords: Cloud computing, Task scheduling, Makespan, Cuckoo search, Crow search.

1. Introduction

Cloud computing could be rightly regarded as the metaphor for the Internet. Cloud provides access from anywhere and at any time. Cloud computing removes the physical barrier for users while providing access to its resources, with users just requiring internet connectivity from their end [1, 2]. Cloud grants access to its resources to the users over the internet as a service. The cloud environment mainly adopts the pay-per-use model while offering its services to the users. There are several cloud service providers who could provide the requested services to the cloud users. Users just need to log-in with their credentials for accessing the cloud services. Cloud users are relieved from storing their data from their own personal computers and instead could do so virtually in the cloud. They can also submit their applications to the cloud and make use of servers available at the cloud for getting their application manipulated and processed. Such characteristics of cloud attract many individuals as well as corporates to make use of the cloud services for executing their

applications. Infrastructure as a service (IaaS), offered by the cloud acts like a foundation for other higher level type of cloud services like the Platform as a Service (PaaS) and Software as a Service (SaaS) [3-7]. Due to its remarkable features like cost effectiveness, reliability and scalability, the cloud has gained a strong foothold in business as well as scientific communities. It relieves the users from making any upfront investment to their infrastructure for executing their applications. Users need to pay for the duration of usage of cloud resources to the cloud service providers without worrying much about the complex IT cloud infrastructure.

The performance of the cloud computing environment primarily depends upon the task scheduling strategy implemented inside it. Scheduling in cloud is a process that maps a bunch of received workload to a bunch of virtual machines that are capable of executing them or alternately, allocating the virtual machines with available resources to meet the user demands. Optimization of resources could be achieved through efficient scheduling that would in turn increase the performance efficiency of the cloud environment. Since there is an enormous increase from the user side approaching the cloud to meet their application demands, there needs to be efficient scheduling algorithms deployed in cloud for efficiently allocating the user tasks or jobs to appropriate data centres in cloud.

2. Related work

In [8] A comprehensive multi-objective algorithm for task scheduling had been proposed. The authors had considered four objectives that are quite conflicting in nature namely cost of task execution, task transfer duration, power consumed and task queue length in their work. The algorithm considered both the costs of execution and power consumption and accounted for its reduction from both customer and provider perspectives. The performance of the proposed algorithm was compared with other multi-objective algorithms like Multi-Objective Particle Swarm Optimization (MOPSO) and Multi-Objective Genetic Algorithm (MOGA). Results obtained proved the superiority of the proposed algorithm.

In [9] An innovative multi-objective Cuckoo Search Optimization (MOCSO) algorithm had been proposed to address the resource scheduling issues in cloud environment. The proposed algorithm minimized the cost incurred by the cloud user and increased the performance of the system by minimizing the makespan. This technique accounted for maximum resource utilization yielding increased profit for the cloud service providers.

In [10] had proposed an algorithm by hybridizing the Cuckoo Search and Harmony Search algorithms named, CHSA for optimizing the scheduling performance in the cloud environment. The Cuckoo search and Harmony search algorithms were integrated to carry out intelligent process scheduling. A multi-objective function comprising of parameters namely cost, energy consumption, memory expended, and credit gained and penalty accrued had been proposed by the authors. The performance of the proposed CHSA had been verified by comparing it with the hybrid cuckoo gravitational search algorithm, Cuckoo search and Harmony search algorithms with respect to various multi-objective parameters.

In [11] had proposed an algorithm by hybridizing the Cuckoo Search and Particle Swarm Optimization algorithms named, CPSO for improvising the scheduling performance in the cloud computing environment. QoS parameters namely makespan, cost and deadline violation rate were effectively reduced by the proposed CPSO algorithm. The proposed CPSO algorithm had been evaluated for its performance using the Cloudsim toolkit. Simulation results obtained had proved the efficiency of the proposed CPSO algorithm

In [12], the authors had allocated the task to a virtual machine that accounted for least execution cost and had the capability to meet the specified deadline constraints. Also, the virtual machine that had been allocated the task was placed under the most utilized physical host category matching to its capability. The authors had evaluated the performance of their proposed co-optimization technique, joint task scheduling and VM placement (JTSVMP), by comparing it with other multi-optimization algorithms. The results obtained had proved that the proposed co-optimization technique had effectively reduced the makespan, execution cost, degree of imbalance and maximized the physical hosts' resource utilization capability.

In [13], the authors had integrated the genetic algorithm and the electro search algorithm and proposed a Hybrid Electro Search with a genetic algorithm (HESGA) for improving the task scheduling process in the cloud. The authors had considered the QoS parameters namely makespan, load balancing, resource utilization and cost for fine tuning the task scheduling activity. Best local optimal solutions were generated using the Genetic Algorithm whereas the Electro Search algorithm generated the best global optimal solutions. Experimental results obtained had shown that the

proposed HESGA outperformed a host of other existing algorithms.

In [14] had integrated the genetic algorithm (GA) and the Bacterial Foraging (BF) algorithm by combining their most desirable characteristics. This had led to the improvement of scheduling performance in the cloud environment. The proposed hybridized scheduling algorithm successfully accounted for the reduction of makespan, energy consumption with respect to both economic and ecological perspectives.

The above literature review does not provide near optimal solution for QoS parameters makespan and cost when considered together. The proposed Cuckoo Crow Search Algorithm (CCSA) considers the makespan and cost parameters for optimizing the task scheduling and resource utilization activity amongst the virtual machines in the cloud environment [15, 16].

The remaining of this article had been structured as follows: The proposed CCSA algorithm for optimizing the task scheduling process had been described in the Section 6.1. Results obtained and their subsequent analysis had been presented in Section 7. Conclusion and future work had been presented in the Section 8.

3. Problem with solution framework

The cloud service providers set up an environment in cloud comprising of the physical machine (PM) and the virtual machine (VM) for providing a public interface. The cloud consumers submit their tasks through the interface. All of such received task requests are aggregated and effectively managed by the Request Manager. The Resource Monitor keeps an update of the availability record of the cloud resource pool that includes CPU, memory and storage. The Scheduler component effectively schedules the tasks in the cloud environment such that the formulated fitness function is minimized. The constrained tasks are assigned to virtual machines in accordance to the latter's performance in the scheduling process. The Scheduler, after obtaining the needed information from both the Request Manager and Resource Monitor, begins to schedule the tasks. Once the needed information is obtained, a decision regarding the allocation of tasks to appropriate virtual machines is made.

Each task needs to be allocated to appropriate virtual machine. The allocation process can be fine-tuned after gathering the location information of the VMs. This information helps in decreasing a host of parameters including the migration cost, total time, load utilization and energy consumption.

Table 1. Notation used in CCSA algorithm

Notations	Description
PM_i	Physical machine i, $1 \le i \le n$
VM_i	Virtual machine i, $1 \le i \le I$
T_i	Task i, $1 \le i \le m$
C_i	Cost
ET	Execution time
BW	Bandwidth
w_1, w_2	Control Parameters
Mips	Million instruction per second

Let T_a represent the task submitted by the user. Each of such received task needs to be assigned to appropriate virtual machines (VM_i). Notations used in the CCSA algorithm are depicted in Table 1 above.

4. Problem statement

A task set T, could be defined as $T = \{t_1, t_2 ... t_m\}$, where m indicates the total number of tasks. The Virtual Machine (VM) set could be defined as $VM = \{vm_1, vm_2 ... vm_n\}$, where n represents the total number of virtual machines. The mapping of a task to any specific virtual machine could be represented in a matrix (M) form as depicted below:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{1n} \\ m_{21} & m_{22} & m_{2n} \\ m_{m1} & m_{m2} & m_{mn} \end{bmatrix} \quad (1)$$

Where m_{ij} represents the assignment association between the i^{th} task on the j^{th} virtual machine. The time expected to execute a task on any specific virtual machine could be represented in a matrix (ET) form as shown below:

$$ET = \begin{bmatrix} et_{11} & et_{12} & et_{1n} \\ et_{21} & et_{22} & et_{2n} \\ et_{m1} & et_{m2} & et_{mn} \end{bmatrix} \quad (2)$$

Here

$$et_{ij} = \frac{t_{length\_i}}{VM_{comp\_j}} \quad (3)$$

Where et_{ij} is the execution time of task i on virtual machine j, t_{length_i} denotes the length of each $task_i$, which is actually the number of instructions that are present in a task that is being executed, $VMcomp\_j$ is the processing power of the virtual machine j and the same could be obtained from the below eqn:

$$VM_{comp\_j} = VM_{mips\_j} \times VM_{penum\_j} \quad (4)$$

Where *VMmips*_j is the computing power of the virtual machine *j* and *VMpenum*_j denotes the number of CPUs that are utilized in virtual machine *j*. The time taken for transmitting the task can be represented in a matrix (ER) form as depicted below:

$$ER = \begin{bmatrix} er_{11} & er_{12} & er_{1n} \\ er_{21} & er_{22} & er_{2n} \\ er_{m1} & er_{m2} & er_{mn} \end{bmatrix} \quad (5)$$

Here

$$er_{ij} = \frac{t_{inputfilesize\_i}}{VM_{bw\_j}} \quad (6)$$

Where er_{ij} represents the time at which the task$_i$ is assigned to j^{th} virtual machine, $t_{inputfilesize\_i}$ indicates the size of $data_i$ for $task_i$ and VM_{bw_j} is the available bandwidth at the j^{th} virtual machine. The time at which an individual task gets completed can be obtained from the equation given below:

$$TaskRunTime_{ij} = et_{ij} + er_{ij} \quad (7)$$

Therefore the time taken for completing the tasks that were assigned to j^{th} virtual machine could be obtained by summing up the time taken to complete all the tasks that were allocated to the j^{th} virtual machine as shown below:

$$VM_{complete\_j} = \sum_{i=1}^{k} TaskRunTime_{ij} \quad (8)$$

Since the virtual machines are subjected to simultaneous execution, the task completion time could then be deduced by appropriating the time taken by the last virtual machine to complete the assigned task [10]. This has been represented as shown below:

$$C_{Time(I)} = \max\left(VM_{completetime\_j}\right) j \in [1, n] \quad (9)$$

Let the unit cost for completing a task at the jth virtual machine be $Ucost_j$, then the total cost for executing all subtasks could be represented as shown below:

$$Totalcost(I) = \sum_{j=1}^{n} VM_{completetime\_j} * Ucost_j \quad (10)$$

The time constraint function for task scheduling activity could be defined as shown below:

$$Time_{cf(I)} = \frac{C_{Time(I)} - C\_Time_{min}}{C_{Time_{max}} - C\_Time_{min}} \quad (11)$$

Where C_Time_{min} is the time consumed by the task while getting executed at the best virtual machine, C_Time_{max} is time consumed by the task while executed at the worst machine. The values of C_Time_{min} and C_Time_{max} could be deduced as shown below:

$$C_{Time_{min}} = \frac{\sum_{i=1}^{m} t_{length\_i}}{n \times \max(VM_{comp\_j})} + \frac{\sum_{i=1}^{m} t_{inputsize\_i}}{n \times VM_{bw\_j}} \quad (12)$$

$$C_{Time_{max}} = \frac{\sum_{i=1}^{m} t_{length\_i}}{n \times \max(vm_{comp\_j})} + \frac{\sum_{i=1}^{m} t_{inputsize\_i}}{n \times VM_{bw\_j}} \quad (13)$$

In a similar fashion the cost constraint function for task scheduling activity could be defined as shown below:

$$C_{Cost(I)} = \frac{TotalCost(I) - TotalCost_{min}}{TotalCost_{max} - TotalCost_{min}} \quad (14)$$

Where $TotalCost_{min}$ indicates the cost expended by the user task while getting executed on a least cost virtual machine in parallel and $TotalCost_{max}$ indicates the cost expended by the user task while getting executed on an expensive virtual machine in parallel. The values of $TotalCost_{min}$ and $TotalCost_{max}$ could be deduced as given below:

$$TotalCost_{min} = C_{Time_{min}} \times MIN\left(UCost_j\right) \quad (15)$$

$$TotalCost_{max} = C_{Time_{max}} \times MIN\left(UCost_j\right) \quad (16)$$

Finally the fitness function for the proposed approach could be formulated as given below:

$$Fitness\ Function = w_1 Tim + w_2 C_{Cost} \quad (17)$$

In the function, $w1+w2=1$, $w1$ is the weight factor of time and $w2$ is the weight factor of the cost.

5. Cuckoo search algorithm

The cuckoo species that hatch their eggs in other host birds' nest had been the inspiration for the Cuckoo Search algorithm. [10] The cuckoo bird generally hatches their fertilized eggs in other cuckoos' nest with the intention of raising their baby birds by other proxy parent cuckoos. Every egg

present in the cuckoo nest denotes a solution, with the cuckoo egg denoting an innovative solution. Once the host birds identify that the eggs are not their own, they either abandon them by throwing them off their nest or just leave their nest once for all and build a new nest for them. The Cuckoo Search optimization algorithm adheres to three basic rules listed below:

1. Every cuckoo hatches a single egg at a time and drops it in a randomly selected nest.
2. The nests containing high quality eggs (solutions) will be carried over to succeeding generations.
3. The available host nests are fixed and at any point of time a host cuckoo bird can very well identify the presence of a foreign egg with the probability p ϵ [0, 1]. In such case, the host cuckoo can either just throw the egg off the nest or leaves its nest once for all and builds its own nest at a new location.

The rule 3 presented above could be tweaked by swapping a fraction (represented as p_a) of the total available n host nests with new nests (infusing new random solutions). The chosen solution is deemed to be qualified or fit when it is proportional to the value of identified objective function. Implementation-wise, the principle adopted is that every egg in a nest represents a solution and a cuckoo can hatch only one egg at a time (which represents a new solution). The intention is to select the new and hypothetically better solution (cuckoo egg) by swapping it with an inferior solution (cuckoo egg) from the nest.

Since the Cuckoo Search algorithm balances the local random walk and global random walk more effectively, it is normally best suited for providing solutions to global optimization problems. Control and balance between the local and global random walks are achieved through the switching parameter $p_a \in [0, 1]$. The local and global random walks can be represented using Eqs. (18) and (19) respectively, as shown below:

$$X_i^{t+1} = X_i^t + \alpha s \otimes H(P_a - \varepsilon) \otimes \left(X_j^t - X_k^t\right) \quad (18)$$

$$X_i^{t+1} = X_i^t + \alpha L(s, \lambda) \quad (19)$$

Where X_i^{t+1} indicates the Next position, X_i^t represents the current positions selected by applying the random permutation, α represents the positive step size scaling factor, s indicates the step size, $\otimes$ indicates the entry-wise product of two vectors, H indicates heavy-side function, the P_a value is used to adjust between local and global random walks, ε is the random number obtained from uniform distribution, $L(s, \lambda)$ indicates the Levy distribution that is applied for defining the step size of random walk.

6. Crow search algorithm

Crows (crow species) are noted for their intelligence [17]. Their brain is as large as their body size when the brain-to-body ratio is taken into consideration, the crow's brain slightly lags behind the human brain. The intelligence characteristics of crow have been strongly established with many a proof. Despite the fact that they lag behind humans in intelligence, they build tools and have the possess the capability of identifying themselves in mirrors. They can recollect faces and alert each other when a foe approaches them. They have the most sophisticated form of communication and can recollect where they had hidden their food even after several months.

Crows keep an eye on other bird species, look for where these birds are hiding their food and eventually steal their food once they find a chance. Whenever a crow commits a theft, it starts moving away to newer locations so that it itself doesn't become a victim of theft in future. Since they themselves had enacted a theft, they clearly read the intention of other thieves and make themselves secured from being robbed.
This population-based behaviour of crows, like storing their extra food in hidden locations and recovering it during their needy times had been incorporated in this article. The principles of Crow Search algorithm are herewith enlisted:

1. Crows live in groups.
2. Crows are capable enough to recollect where they had dumped their food.
3. Crows trail each other and whenever they find a chance, steal other's food.
4. Crows safeguard their location from adversaries with a probability in the interval [0, 1].

Let us assume an environment with d-dimension that includes a number of crows. Let N be their flock size and the location of the $crow_i$ at time (iteration) iter, in the search space is represented by a vector denoted by X^{iter}, and *i=1, 2...*, *N*.

Where $iter = 1,2,..iter_{max}$, $X^{iter} = [X_1^{iter}, X_2^{iter}, ...X_d^{iter}]$ and $iter_{max}$ indicates the maximum number of iterations.

Each crow recollects the location of its hiding place from its memory. At iteration *iter*, the location of hiding of $crow_i$ could be given by m^{iter}. This location is the best one which the $crow_i$ had obtained

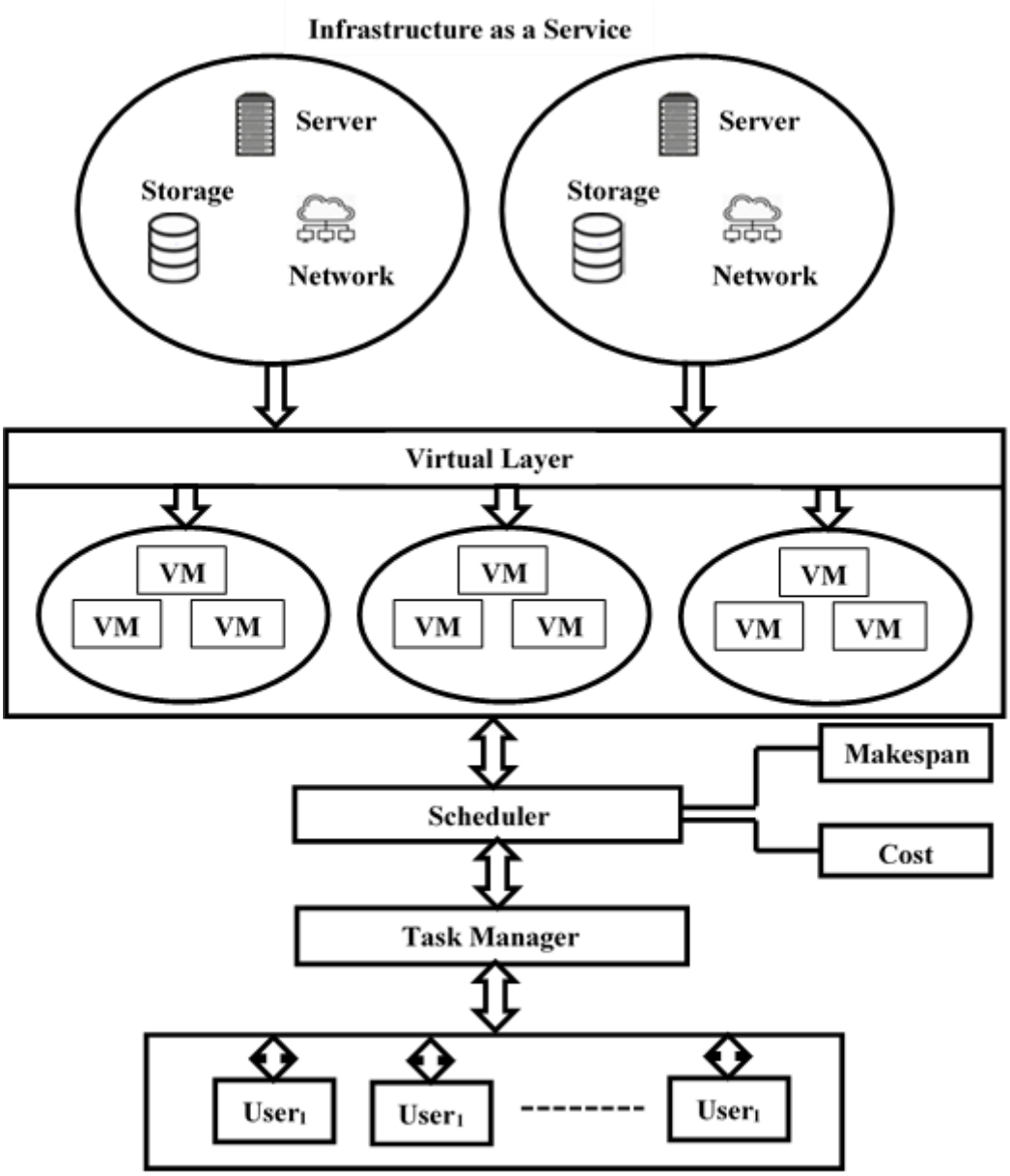

Figure. 1 Architecture of the proposed hybrid cuckoo crow search algorithm

till now. In fact, each crow memorizes the location that it considers to be the best one. Crows roam around the environment in search of better food sources.

Let us consider that at iteration *iter*, $crow_j$ visits its hiding location m^{iter}. Also, in the same iteration, $crow_i$ follows $crow_j$ to identify its hiding location. At this juncture two scenarios are possible:

Scenario 1: $Crow_j$ is unaware of it being followed by $crow_i$. Subsequently the hidden location of $crow_j$ is revealed to $crow_i$.

Scenario 2: $Crow_j$ is aware of it being followed by $crow_i$. Usually in such case, crows fly to a different place far much far away from the place where they had hidden their food, just to fool their followers.

Now both scenario 1 and 2 could be together represented as given below:

$$X^{iter+1} = \begin{cases} X^{iter} + r_i \times fl^{iter} \times \left(m^{iter} - X^{iter}\right) & \text{if } r_i \geq AP^{j,iter} \\ a\ random\ position & otherwise \end{cases} \quad (20)$$

Where r_i is a random number that is randomly distribution between 0 and 1 and fl^{iter} represents distance the crow flies, $AP^{j,iter}$ indicates the awareness probability of $crow_j$, at iteration *iter*.

6.1 Task Scheduling using Hybrid CCSA

The main objective of the proposed approach is to optimize the task scheduling process by applying

the hybridized Cuckoo Crow Search Algorithm (CCSA).

Pseudo code of CCSA

Let the position of crow population be initialized to $xi = (x_i1, x_i2, \ldots x_in)$
Consider an n dimensional search space.
Let $crow_i$ denotes the crow population where $i = 1,2, \ldots, N$.
Initialize the initial memory location (m_i) of each crow to $m_i = m_i1, m_i2, \ldots m_in$
Each crow is evaluated by considering the quality of its position by applying in the fitness function that had been formulated.
Iteration, iter begins at 1
New location for each crow, inside the search space could be generated as follows:

Let any $crow_i$ select randomly any other $crow_j$ from the crow family.
The $crow_i$ follows $crow_j$ to find the location where it has hidden its food.
AS long as iter remains lesser than max number of iterations,
Then, for each crow
Update its new location in accordance and also
Update the new location inside its memory according to Eq. (20)
Update using CS according to Eq. (19)
iter = iter + 1
End.

The overall architecture of proposed Hybrid Cuckoo Crow Search algorithm for efficient task scheduling had been depicted in Figure 1. The tasks submitted by the users are handed over to the Task Manager. The Task Manager relays these tasks to the Scheduler component that schedules them based on their fitness function. The Resource component tracks the utilization of virtual machines with respect to the QoS parameters memory, energy and CPU. The proposed Hybrid Cuckoo Crow Search Algorithm optimizes the task scheduling process by reducing the cost and execution time[18-22].

7. Results and discussion

Experiments were conducted in a simulated environment using Java (jdk 1.6) with Cloudsim tool. The implementation setup comprised of a PC with Windows 7 OS @ 2 GHz dual core, a RAM of 4 GB and a 64-bit windows 2007 OS. The experiments were conducted by varying the input task from 50 to 300 numbers. The performance of the proposed Hybrid CCSA had been evaluated by comparing its results with respect to cost and execution time (makespan) parameters against MO-ACO, ACO and MIN-MIN algorithms [23]. Both ACO and Min-Min algorithms aim to provide optimized solutions for task scheduling activity in cloud by continuously iterating the candidate solutions. In a similar manner, the MO-ACO algorithm generates optimized solution for task scheduling activity by taking into consideration of makespan and cost parameters. Since the proposed CCSA also strives to provide a near optimal solution to task scheduling activity, it has been compared with the MO-ACO, ACO, Min-Min algorithms. The Table 2 lists out the types of entity, parameters and their corresponding values that are considered in the experiment.

Table 2. Parameter setting for Cloud Simulator

Entity Type	Parameters	Values
Task (Cloudlet)	Task Length	5000-100000
	Total tasks	50-250
	File size	300-5000
Virtual Machine (VM)	MIPS	512-1024
	Number of VM	10
	Bandwidth	500-1200
	Memory	512-2048
	Storage	100000-800000
	Unit cost of VM	1-10

7.1 Comparison of makespan

The proposed CCSA increases the efficiency of the task scheduling process by minimizing the makespan and cost parameters. The performance of the proposed CCSA with respect to the makespan parameter had been compared with MO-ACO, ACO and MIN-MIN algorithms. In Figure 2, the comparison of task execution time using 10 VMs by applying CCSA, MO-ACO, ACO and MIN-MIN algorithms had been shown. For 50 number of tasks, the makespan values obtained are 150.52,165, 174.69 and 183.706 for CCSA, MO-ACO, ACO and MIN-MIN respectively. When the task count is increased to 100 numbers, the makespan values obtained are 315.4, 332.347, 358.802 and385.199 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 150 tasks, the makespan values obtained are 500.89, 516.45, 542.876 and 560.606 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 200 tasks, the values obtained are 652.78, 674.495, 726.978 and 770.794 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 250 tasks, the values obtained are 780.8, 797.702, 850.157 and 981.011 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 300 tasks, the values obtained are 970,990.471,

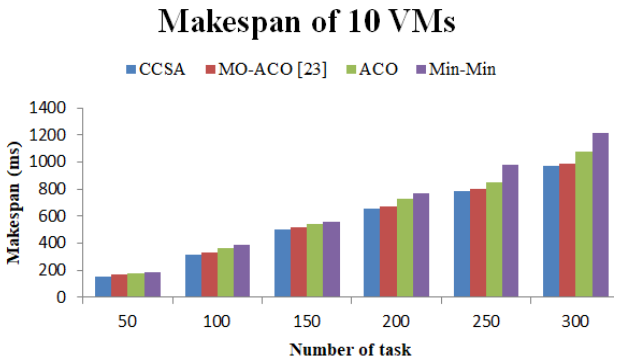

Figure. 2 Makespan of 10 VMs

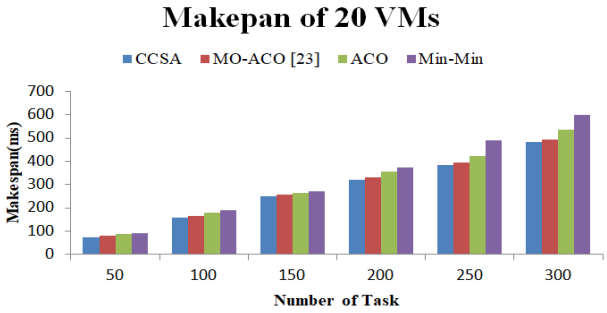

Figure. 3 Makespan of 20 VMs

1077.82 and 1217.23 for CCSA, MO-ACO, ACO and MIN-MIN respectively. On the whole it could be inferred that the proposed CCSA is superior to other algorithms while comparing the makespan values deduced using 10 VMs.

In Fig. 3, the comparison of task execution time using 20 VMs by applying CCSA, MO-ACO, ACO and MIN-MIN algorithms had been shown. For 50 number of tasks, the makespan values obtained are 74.4, 80.8, 86.5 and 90.5 for CCSA, MO-ACO, ACO and MIN-MIN respectively. When the task count is increased to 100 numbers, the makespan values obtained are 156.7, 165.4, 178.5, 191.5 and 385.199 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 150 tasks, the values obtained are 248.5, 256.4,265 and 270.7for CCSA, MO-ACO, ACO and MIN-MIN respectively.

For 200 tasks, the values obtained are 320.4, 330.5, 356 and 375 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 250 tasks, the values obtained are 385.1, 395.7, 423.5 and 488.445 for CCSA, MO-ACO, ACO and MIN-MIN respectively. For 300 tasks, the values obtained are 482.5, 492.5, 535.4 and 600.8 for CCSA, MO-ACO, ACO and MIN-MIN respectively. Hence it could be inferred that the proposed CCSA is superior to other algorithms while comparing the makespan values deduced using 20 VMs.

7.2 Comparison of cost

The performance of the proposed CCSA with respect to the cost parameter had been compared with MO-ACO, ACO and MIN-MIN algorithms. Figure 4 shows the cost expended for tasks numbered 50, 100, 150, 200, 250 and 300 using 10 VMs by applying the CCSA, MO-ACO, ACO and MIN-MIN algorithms. It could be seen that cost values obtained for 50 tasks are 5400, 5590.28, 5833.3 and 5900.1 for CCSA, MO-ACO, ACO and MIN-MIN respectively. The cost values in the case of 100 tasks are 5400, 5590.28, 5833.3 and 5900.1 for CCSA, MO-ACO, ACO and MIN-MIN respectively. In the case of 150 tasks, the values obtained are 15000.78, 6284.7, 17986.1 and 19930.6 for MO-ACO, ACO and MIN-MIN respectively. For 200 number of tasks, the cost values are 22650.8, 22847.2, 24305.6 and 26250 for MO-ACO, ACO and MIN-MIN respectively. For 250 number of tasks, the cost values are 27500, 28437.5, 30381.9 and 33541.7 for MO-ACO, ACO and MIN-MIN respectively. The cost for 300 tasks is 30000, 31500.4, 32513.4 and 33450.7for MO-ACO, ACO and MIN-MIN respectively. On the whole it could be inferred that the proposed CCSA is superior to other algorithms while comparing the cost values deduced using 10 VMs.

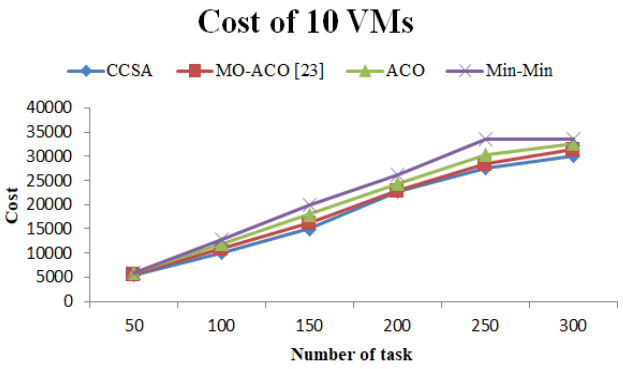

Figure. 4 Cost of 10 VMs

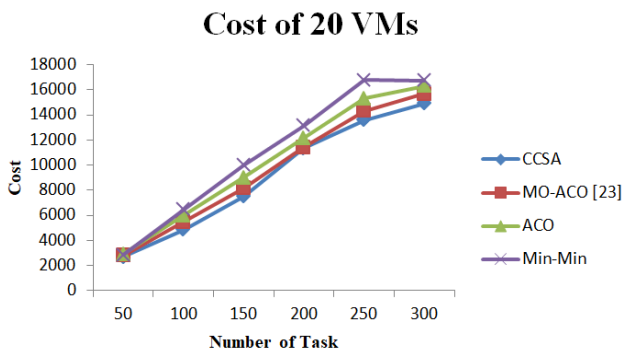

Figure. 5 Cost of 20 VMs

Fig. 5 shows the cost expended for tasks numbered 50, 100, 150, 200, 250 and 300 using 20 VMs by applying the CCSA, MO-ACO, ACO and MIN-MIN algorithms. It could be seen that cost values obtained for 50 tasks are 2705.2, 2793.34, 2925.32 and 2840.67 for CCSA, MO-ACO, ACO and MIN-MIN respectively. The cost values in the case of 100 tasks are 4855.7,5468.75,5987.4 and 6452.89 for CCSA, MO-ACO, ACO and MIN-MIN respectively. In the case of 150 tasks, the values obtained are 7489.8, 8142.35, 8993.05 and 9978.2 for MO-ACO, ACO and MIN-MIN respectively. For 200 number of tasks, the cost values are 11358.78, 11420.8, 12158.9 and 13128 for MO-ACO, ACO and MIN-MIN respectively. For 250 number of tasks, the cost values are 13580, 14325.9, 15296 and 16770.85 for MO-ACO, ACO and MIN-MIN respectively. The cost for 300 tasks is 14890.4, 15688, 16300.78 and 16749.25 for MO-ACO, ACO and MIN-MIN respectively. Hence it could be inferred that the proposed CCSA is superior to other algorithms while comparing the cost values deduced using 20 VMs.

8. Conclusion and future work

The hybrid CCSA had been proposed to address the task scheduling issue in cloud computing. The hybrid CCSA takes into consideration the QoS parameters makespan and cost for increasing the performance of the task scheduling activity. Experimental results obtained using hybrid CCSA had been compared with MO-ACO, ACO and Min-Min algorithms. Results obtained had shown that the proposed hybrid CCSA had produced better output with respect to makespan and cost parameters compared to other algorithms. For experimental analysis, the task count was varied from 50 to 300 number and 10 and 20 numbers of Virtual Machines were considered. By analysing the results obtained, it had been found that the proposed CCSA technique had produced an improvement of 3.14%, 10.70%, and 21.60% for makespan and had reduced the overall cost to the tune of 4.56%, 11.19% and 19.35% when compared with MO-ACO, ACO, Min-Min algorithms respectively when used with 10 VMs. By integrating the Cuckoo Search and Crow Search algorithms, a highly efficient solution for task scheduling issue in cloud computing had been devised. As a future activity, more QoS parameters could be taken into consideration for testing the efficiency of the proposed hybrid CCSA and the same could be evaluated under real time scenarios.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

"Conceptualization, Pradeep Krishnadoss; methodology, Javid Ali ; software, Javid Ali; validation, Gobalakrishnan Natesan; formal analysis, Gobalakrishnan Natesan; investigation, Manikandan Nanjappan; resources, Manikandan Nanjappan; data curation, Manikandan Nanjappan; writing— Pradeep Krishnadoss; writing—review and editing, Pradeep Krishnadoss; visualization, Parkavi Krishnamoorthy; supervision, Vijayakumar Kedalu Poornachary".

References

[1] 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.

[2] 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.

[3] S. Elsherbiny, E. Eldaydamony, M. Alrahmawy, and A. E. Reyad, "An extended Intelligent Water Drops algorithm for workflow scheduling in cloud computing environment", *Egyptian Informatics Journal*, Vol. 19, No. 1, pp.33-55, 2018.

[4] C. W. Tsai, W. C Huang, M. H. Chiang, M. C. Chiang, and C. S. Yang, "A hyper-heuristic scheduling algorithm for cloud", *IEEE Transactions on Cloud Computing*, Vol. 2, No. 2, pp. 236-250, 2014.

[5] Z. Tang, L. Jiang, J. Zhou, K. Li, and K. Li, "A self-adaptive scheduling algorithm for reduce start time", *Future Generation Computer Systems*, Vol. 43, pp. 51-60, 2015.

[6] B. Tripathy, S. Dash, and S. K. Padhy, "Dynamic task scheduling using a directed neural network", *Journal of Parallel and Distributed Computing*, Vol. 75, pp. 101-106, 2015.

[7] H. Topcuoglu, S. Hariri, and M. Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, No. 3, pp. 260-274, 2002.

[8] F. Ramezani, J. Lu, J. Taheri, and F. K. Hussain, "Evolutionary algorithm-based multi-objective task scheduling optimization model in cloud environments", *World Wide Web*, Vol. 18, No. 6, pp. 1737-1757, 2015.

[9] S. H. H. Madni, M. S. Abd Latiff, and J. Ali, "Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds", *Arabian Journal for Science and Engineering*, Vol. 44, No. 4, pp. 3585-3602, 2019.

[10] K. Pradeep and T. P. Jacob, "A Hybrid Approach for Task Scheduling Using the Cuckoo and Harmony Search in Cloud Computing Environment", *Wireless Personal Communications*, Vol. 101, No. 4, pp 2287–2311, 2018.

[11] K. Pradeep and T. P. Jacob, "A multi-objective optimal task scheduling in cloud environment using cuckoo particle swarm optimization", *Wireless Personal Communications*, Vol. 109, No. 1, pp. 315-331, 2019.

[12] A. Dabiah, H. Tianfield, Y. Zhang, and B. Pranggono, "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers", *Future Generation Computer Systems*, Vol. 115, pp. 201-212, 2021.

[13] S. Velliangiri, P. Karthikeyan, V. A. Xavier, and D. Baswaraj, "Hybrid electro search with genetic algorithm for task scheduling in cloud computing", *Ain Shams Engineering Journal*, Vol. 12, No. 1, pp. 631-639, 2021.

[14] S. Srichandan, T. A. Kumar, and S. Bibhudatta, "Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm", *Future Computing and Informatics Journal*, Vol. 3, No. 2, pp. 210-230, 2018.

[15] P. Krishnadoss and P. Jacob, "OCSA: Task Scheduling Algorithm in Cloud Computing Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 3, pp. 271-279, 2018.

[16] K. Pradeep and T. P. Jacob, "CGSA scheduler: A multi-objective-based hybrid approach for task scheduling in cloud environment", *Information Security Journal: A Global Perspective*, Vol. 27, No. 2, pp. 77-91, 2018.

[17] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems", *Crow Search Algorithm Computers & Structures*, Vol. 169, pp. 1-12, 2016.

[18] N. Gobalakrishnan and C. Arun, "A New Multi-Objective Optimal Programming Model for Task Scheduling using Genetic Gray Wolf Optimization in Cloud Computing", *The Computer Journal*, Vol. 61, No. 10, pp. 1523-1536, 2018.

[19] G. Natesan and A. Chokkalingam, "Opposition Learning-Based Grey Wolf Optimizer Algorithm for Parallel Machine Scheduling in Cloud Environment", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 1, pp.186-195, 2017.

[20] K. Pradeep and T. P. Jacob, "Comparative analysis of scheduling and load balancing algorithms in cloud environment", In: *Proc. of International Conf. on Control, Instrumentation, Communication and Computational Technologies*, pp. 526-531, 2016.

[21] G Natesan and A Chokkalingam, "Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment", *Wireless Personal Communications*, Vol. 110, pp. 1887-1913, 2020.

[22] 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.

[23] Q. Guo, "Task scheduling based on ant colony optimization in cloud environment", In: *Proc. of AIP Conf proceedings*, pp. 040039, 2017.