Received: June 26, 2021. Revised: July 20, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An H-CSO Algorithm for Workflow Scheduling in Heterogeneous Cloud Environment

Jai Bhagwan[1]* **Sanjeev Kumar**[1]

[1]*Department of Computer Science & Engineering,*
Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India
* Corresponding author's Email: jaitweet@gmail.com

Abstract: Cloud computing technology is becoming popular in both academia and industries these days because of its dynamic and flexible infrastructure which provides good computing facilities through web networks without a place specification. In order to process the workflows, the cloud technology uses datacenters which consist of hosts having various homogeneous and heterogeneous virtual machines. Virtual machines play the role of real machines with multiple operating systems environments. Due to a wide acceptance of the cloud, the load on the cloud is increasing regularly. So, scheduling plays a key role to manage a machine load by mapping several workflows with available virtual machines. The scheduling can be divided into two types: dynamic and static scheduling. Dynamic scheduling can be done effectively by meta-heuristic computing techniques. In the field of cloud computing, many scientists have developed various algorithms for workflow scheduling in order to schedule the workflows. The Cat Swarm Optimization works fine as compared to the Max-Min algorithm, Particle Swarm, and Ant Colony Optimization algorithms. In this research, a new algorithm is designed named H-CSO by using the concept of Heterogeneous Earliest Finish Time and Cat Swarm Optimization algorithms. After experiments, it is found that the proposed H-CSO algorithm gives efficient makespan at realistic costs as matched to the Cat Swarm Optimization. The newly designed H-CSO algorithm is 2.99%, 2.87%, 3.35%, and 5.77% efficient for CyberShake_1000, Montage_1000, Inspiral_1000, and Sipht_1000 datasets respectively as compared to the standard Cat Swarm Optimization in terms of average makespan reduction. In case of cost reduction, the proposed algorithm is 4.03%, 5.12%, 2.42%, and 3.93 effective as compared to the Cat Swarm Optimization for CyberShake_1000, Montage_1000, Inspiral_1000, and Sipht_1000 datasets respectively.

Keywords: Ant colony optimization (ACO), Cloud computing, Cat swarm optimization (CSO), Datacenter, heterogeneous earliest finish time (HEFT), H-CSO, Max-min, Particle swarm optimization (PSO), Self-motivated inertia weight (SMIW), Virtual machines (VMs).

1. Introduction

Nowadays, cloud computing acceptability is growing regularly among various organizations due to its dynamic infrastructure and pay-per-usages flexibility. Cloud computing allows its users to share hardware resources, software applications, network resources, workspace for application developments, mailing services, etc. Cloud technology is an abstraction and isolation of physical resources which works basically on three prototypes namely Software as a Service, Platform as a Service, and Infrastructure as a Service [1, 2]. To fulfil the requirements of a user, cloud technology follows a process of creating a datacenter consisting of a single or various computing machines called hosts connected by high-speed networks. A host is consisting of various homogeneous or heterogeneous virtual machines which act just like physical computing machines having guest operating systems for providing all computing facilities [3, 4]. The heterogeneous environment is a collection of various virtual machines having different computing speeds, memories, bandwidths, and other parameters compared to each other whereas the homogeneous environment is made of various virtual machines having similar parameters. The resources of a host are shared among various virtual machines and a virtual machine can have multiple applications and operating systems that are being used to ensure better resource utilization and optimization for a large group of tasks or workflows [5].

Some issues available in cloud computing are scheduling mechanisms, data-centers network expansion, security, energy consumption, and load balancing [6, 7]. Among these issues, workflow scheduling or resources mapping plays a significant role in cloud resources performance. In workflows scheduling, the workflows tasks sent by a user are scheduled on virtual machines in a manner that the submitted tasks could be executed in a minimum execution time. Such a process has to follow a few constraints which are given by the cloud system for several user requests. These constraints can be satisfied by many parameters such as Makespan, Resources Cost, CPU Utilization, Response Time, Throughput, Waiting Time, Turnaround Time, and lots more [8]. Appropriate scheduling allows processing a large volume of workflows with a specific volume of resources in a minimum time. Many scientists have proposed various algorithms for scheduling workflow tasks in cloud computing system. Each algorithm has its strengths and weaknesses. The objective of the scheduling is to process a large volume of workflow tasks in a minimum makespan and better management of virtual machines under the minimum utilization of processing cost [9].

In this research, the HEFT algorithm has been combined into the Cat Swarm Optimization algorithm. A Self-Motivated Inertia Weight has also been included in the velocity calculation formula of the tracing mode of the CSO to control the outranged velocity problem of the Cat Swarm Optimization. The proposed algorithm achieves better results than Max-Min, ACO, PSO, and CSO algorithms in terms of makespan and computing cost.

1.1 Scheduling algorithms

Workflow tasks or independent task scheduling acts as a key player in the cloud computing system while allocating workflow tasks to virtual machines for computations. As said earlier, the goals of a scheduling algorithm are considered as minimization of makespan, faster response time, increasing resource utilization, managing the system load, and reducing the processing cost, etc [10].

The algorithms used in this paper for designing a new technique are summarized as below:

a) Max-Min – Max-Min algorithm works on the policy of maximum expected completion time (MET). This method is very similar to Min-Min, except it discovers the MCT (maximum expected completion time) of each task available in the Meta-Task list and assigns it to the corresponding resource. As the name suggested, it gives higher importance to large tasks first for processing as compared to smaller tasks. This algorithm gives a good performance when several small tasks are available along with large tasks [11].

b) ACO – Ant Colony Optimization works on the pattern of the behavior of real ants that try to find a straight route among their colonies and food. It was introduced by Dorigo in the year 1992. Ants release pheromone on the ways they move while walking through their colonies and the source of the food. As many ants walk through and drop the evaporation of pheromone, the intensity of pheromone is increased. As time passes on, the more intensity of pheromone helps the ants to find the shortest track between their food sources. The ACO can be useful to resolve various difficulties in many fields as well as task scheduling and load balancing in the area of the cloud system [12]. For scheduling tasks in cloud computing, the number of ants is taken less or equivalent to the number of tasks. To execute the task, every ant starts with an random task T_i and resource $R_{j.}$ The probability function given in Eq. (1) is used to decide a particular task to be executed on a specific resource.

$$P_{ij} = \frac{(\tau_{ij})^{\alpha}\,(\eta_{ij})^{\beta}}{\Sigma(\tau_{ij})^{\alpha}\,(\eta_{ij})^{\beta}} \quad (1)$$

Where, τ_{ij} Symbolises the pheromone value related to task T_i and resource R_j
η_{ij} Represents the heuristic function
α Regulates the influence of pheromone value
β Governs the impact of heuristic function

Here, every ant shapes the entire solution of mapping all tasks to available resources step by step. Initially, the pheromone value is set as a positive constant then, at the end of each iteration number, this value is changed by the ant. Finally, an optimum solution is provided.

c) PSO – Particle Swarm Optimization was announced by Kennedy and Eberhart in 1995 by the inspiration of particles' social behaviours. It is similar to the behaviours of the birds searching for food. The PSO works in the following steps: Initialization of the population, updating the particles' velocity and position, Fitness calculation, and finding out the optimal solution. In the case of search-space having multi-dimensions, each particle is moved and associated with a velocity and its position. The particle finds out its best position based on its experiences as well as its neighbour experiences [13].

d) Cat Swarm Optimization – In 2006 a novel algorithm was developed named Cat Swarm Optimization by the inspiration of PSO and ACO behaviours. This technique copies the behaviour of the actual cat. The authors developed two processes of CSO algorithm: Seeking Process and Tracing Process. The seeking method is said to be a global search and the tracing method is a local search. In seeking mode, the cat takes a rest and moves slowly for seeking the prey. In tracing mode, the cat moves fast with a velocity and few other parameters. There are certain parameters in CSO like Seeking Memory Pool (SMP), Seeking Range of the Selected Dimension (SRD), Self Position Consideration (SPC), Counts of Dimension to Change (CDC), and MR (Mixing Ratio). In SMP, one cat is randomly selected for moving to the next position. CDC and SRD decide the random mutations for new positions. The CDC decides the number of dimensions to be mutated and the amount of mutation is decided by SRD. SPC Boolean value will decide how many replicas can be generated in seeking mode [14].

e) HEFT Algorithm – The HEFT works in two phases. The first phase is task prioritization and the second one is virtual machine selection. In the first phase, HEFT calculates the rank of each task based on average computation time, and average communication cost and assigns priority to each task. The task queue is set by decreasing the order of the task's rank. In the second phase, the tasks are scheduled on the virtual machine which is having Earliest Finish Time [15].

The rest of the paper is drafted as: The related work has been drawn in section 2. Problem formulation is summarized in section 3. The proposed methodology is well exposed in section 4. Section 5 is having a simulation setup, describing simulation parameters, cost plan, and performance metrics. Simulation results and discussion is explained in Section 6. Finally, section 7 comprises the conclusion and future scopes of this research.

2. Related work

In cloud technology, extensive research has taken place in the field of task scheduling and balancing of the load. Several algorithms have been used for task or workflow scheduling with different scheduling policies and scenarios. A deep study is illustrating that in [16] the authors proposed an algorithm named EMM (Enhanced Max-Min). The proposed EMM algorithm was implemented in the CloudSim tool. After simulation, it is found that the proposed EMM algorithm is effective than the standard Max-Min method. The average waiting time and completion time of the scheduling were reduced and resources allocation was optimized. In [17] the scientists proposed a modified Max-Min algorithm. The modified approach is prepared using RASA and Max-Min algorithm strategy. The proposed algorithm works based on expected execution time rather than the completion time. So, the authors are saying the proposed Max-Min method is effective than the original Max-Min in terms of makespan. The scientist [18] improved Ant Colony Optimization using a dynamic volatile coefficient and a virtual machine load weight coefficient in order to improve the searching capacity. The proposed method improved load balancing, convergence speed, and completion time. The proposed algorithm showed its effectiveness as compared to other methods after simulation. The author of [19] introduced a scheduling scheme based on a Modified ACO algorithm named MACO. The algorithm was introduced to achieve multiple objectives. The MACO improved the results by decreasing makespan and degree of imbalance as compared to traditional ACO. The experiments were carried out in the CloudSim tool. The paper [20] showed that a multi-objective hybrid algorithm was developed using Genetic Algorithm and Ant Colony Optimization algorithms named HGA-ACO. The proposed algorithm worked for response time, completion time, and throughput. The pheromone was generated by GA and fed to ACO and the ACO was used to improve the crossover solution of GA. The simulation environment was created using the CloudSim tool. After simulation, the authors found that the HGA-ACO performed efficiently as compared to ACO and GA algorithms. In the paper [21], the major aim of the author was to reduce the makespan and cost of the task scheduling. The author improved the initialization problem of the ACO algorithm and proposed an MO-ACO algorithm. The pheromone update function and the heuristic function were also improved. The CloudSim tool was used for simulation and found that the proposed MO-ACO functioned efficiently as matched to traditional ACO and Min-Min algorithms in respect to the degree of imbalance, computing cost, and makespan. In [22] for allocation of the tasks to virtual machines efficiently, the scientists developed a novel algorithm called SACO. The proposed policy works based on slave ants to improve the global searching power of the Ant Colony Optimization. For improvement, diversification and reinforcement policies were adopted and a long path was avoided for ACO ants. The novel algorithm SACO worked effectively as compared to the ACO and IACO in the account of the makespan. The researcher in [23] introduced an Improved PSO algorithm called IPSO for resource scheduling. The improvement was made based on the changes in the constant coefficients in the velocity variation. The IPSO outperformed the standard PSO in terms of processing time and fitness. The paper [24] described that the authors developed an IPSO algorithm for task allocation problems. The tasks were divided into batches dynamically. The results of the small batches were obtained and merged into a single solution. This process was found effective as compared to the Honey Bee, Round-Robin, and ACO algorithms for load balancing, minimizing the makespan and degree of imbalance. The CloudSim simulator was used for experiments. The authors in [25] proposed an algorithm named Dynamic Multi-Objective Orthogonal Taguchi-Cat. The Taguchi approach was inserted to make a balance between local and global search. The Pareto dominant scheme was also used for selecting the appropriate services by a customer. The proposed DMOOTC was found efficient than Improved Min-Min, MOACO, and MOPSO concerning the execution time and computation cost for the overall environment. In the paper [26], the scientists presented a Cat Swarm Optimization Grounded policy for mapping the tasks on suitable resources. The customized CSO algorithm was tested using a workflow and found efficient as compared to the PSO algorithm in terms of optimal task scheduling, load distribution, Transmission Cost, etc. The reason for the good performance is the position update with intelligence instead of random updating. In [27], the authors modify the traditional Cat Swarm Optimization with Linear Descending Inertia Weight equation in the tracing mode. The idea was implemented using the CloudSim tool and it was found that the designed algorithm CSO-LDIW performed better as linked to PSO-LDIW and the traditional CSO in terms of makespan.

It is identified from the related work that the Max-Min is a heuristic algorithm and used for static scheduling only. Meta-heuristic algorithms provide optimal solutions as compared to the heuristic approaches. Ant Colony Optimization has beaten by the PSO, the Cat Swarm Optimization worked well as compared to the PSO. It is found that the scheduling algorithms have various limitations. For example, the Max-Min algorithm is very old and is not able to deal with dynamic scheduling effectively. The Ant Colony Optimization is best suited for local search and has a slow convergence speed, so it doesn't give good performance in cloud computing. The PSO and CSO algorithms are famous for global searching. The Cat Swarm Optimization algorithm gets reached outside the search space due to the unbalanced velocity calculation formula in the tracing mode. The Cat Swarm Optimization algorithm also gets stuck in local minima because all the time, the maximum number of cats resides in the Seeking Mode.

3. Problem formulation

Task scheduling on virtual machines is a stimulating task in cloud computing. To the current day, various algorithms have been developed by scientists to solve this problem. It is observed from the literature survey that each meta-heuristic algorithm has few limitations. The Max-Min algorithm is useful for static scheduling, and where the tasks are independent. The Max-Min algorithm is very old and starts to map the tasks in decreasing order by their length. So, the performance cannot be achieved well enough. The Ant colony optimization works on the principle of insects i.e. ants. It is a local searching technique; so, its global searching part is weak. Particle Swarm Optimization works on the principle of a flock of birds. This algorithm has very good strength in a case of global searching. So, it easily gets trapped in local minima. The next famous algorithm in the literature is Cat Swarm Optimization which

Table 1. Acronyms

Acronym	Meaning
Cat_K	Current Cat
DAG	Directed Acyclic Graph
DC	Datacenter
exp	Exponential Expression
itr	Current Iteration
itr_{max}	Number of Iterations
MR	Mixing Ratio
$rank_U$	Upward Rank
Rs.	Indian Currency
SMIW	Self-Motivated Inertia Weight
Sec	Seconds
SMP	Seeking Memory Pool
T_i	Current Task
Tn	N number of Tasks
VMs	Virtual Machines
Vm	M number of Virtual Machines
V	Velocity Factor
V_K	Velocity of Current Cat
X_K	Current Position of Cat
X_{BEST}	Best Cat as per Fitness Value
γ	SMIW Method
γ_{max}	Constant Value

works just like the behaviour of a real cat. Most of the Cats are remaining in seeking modes all time. So, it gets trapped in local mode i.e. tracing mode. Sometimes, Cat Swarm Algorithm escapes itself from the search space due to an unbalanced velocity equation in tracing mode, which affects the performance of the algorithm. Hence, a new algorithm needs to be designed to reduce the makespan and processing cost.

Table 1 is showing the meanings of the acronyms used in this research paper.

4. Proposed methodology

It is found from the literature review that every algorithm alone is not able to perform the optimal solution for workflow scheduling. Many researchers suggested that improved versions of algorithms are required due to many weaknesses like local minima trapping, pre-mature convergence, escaping of the CSO from search space, etc. Although CSO (Cat Swarm Optimization) works fine in terms of computation time and convergence still, it has some limitations which are identified by literature review and described in detail as follows:

1) Initial populations of cats are given randomly, which increases the computation time.
2) Sometimes Cats move outside the search space due to an unexpected jump in the velocity of the Cats. This disturbs the performance of CSO.
3) The global search process of CSO is quite fine but without a better local search, it could be difficult to find optimal results. The Cat Swarm Optimization algorithm gets jammed in local optima due to the number of cats in the seeking procedure is always remaining more than the cats residing in the tracing procedure. This may affect the mutation process in the tracing mode due to which desirable optimal results could not be obtained.
4) There is a balance mismatch between Tracing and Seeking Mode. Due to which the computation time may increase while working with this algorithm.

In this research paper, a new algorithm has been introduced using a combination of HEFT, and CSO to make efficient the performance of the cloud system. The Self-Motivated Inertia Weight (SMIW) factor has also been incorporated to overcome the problem of outranging the Cats' velocity. Hence, there is no need to check and set the velocities to their initial values again and again and the performance may be increased. The SMIW factor is calculated by Eq. (3) explained in step 4.

The description of the proposed H-CSO algorithm is given in the following steps:

Step 1: Initialize various parameters like velocity factor (V) for all cats. For position update, c, and γ_{max} are set greater than 1 let's say 2.0 value for each. The coefficient c1 is set to 1.5. Random variable r1 varies between 0 and 1. The number of iterations is 300. The numbers of initial solutions are set to 100. These parameters are also shown in Table 2. MR (Mixing Ratio) factor is set to 0.2-0.3 which means that 20%-30% of Cats will be distributed to the Tracing mode and the remaining will be in the Seeking mode of the CSO algorithm.

Step 2: The average execution times of all VMs are calculated. After this, the workflows are fed into the HEFT algorithm. In the case of a workflow, the HEFT policy sets the ranks upwards i.e. from children to parents. If the current task is the last task in the DAG then the rank is calculated by the average execution of all VMs otherwise the HEFT technique sets the ranking to the workflow tasks by the following Eq. (2).

$$rank_u(t_i) = WAvg_i + \\ Max\ t_j \epsilon\ Succ\ (t_i) \left(CAvg_{ij} + rank_u(t_j)\right) \qquad (2)$$

Where $WAvg_i$ is the average execution cost, Succ (t_i) is the set of the immediate successor of task t_i, and $CAvg_{i,j}$ is the average communication cost.

After assigning the workflows tasks, the tasks are submitted to VMs for processing. Then, the solution is checked by the fitness function given in Eq. (10). If the solution is optimized then, the algorithm is stopped otherwise it goes to the next step.

Step 3: In this step, the population obtained by the HEFT policy is filled into the CSO algorithm. Here, S numbers of Cats are created with the help of the HEFT population. The initial velocity values are given to each cat in dimension D. After this, by using the MR value the cats are dispersed into seeking and tracing steps.

Step 4: The current Cat_K is checked for seeking or tracing process. If it is found in seeking place then, the seeking method is processed otherwise the tracing method is processed. The seeking and tracing modes are explained as below:

a) Seeking Mode: In seeking mode the positions of the current Cat_K is replaced by the following way: The S numbers of copies are generated of current Cat_K as per SMP. Now, it is checked the fitness of generated copies of Cat_K and many best cats are found. The best cat is picked up randomly and the current

Received: June 26, 2021. Revised: July 20, 2021.

```
Proposed Algorithm (H-CSO)
Input (Tasks ($T_1, T_2, T_3 \ldots T_n$), Virtual Machines ($VM_1, VM_2, VM_3 \ldots VM_m$)
Output (Optimal Makespan and Cost of n Tasks on m VMs)
BEGIN PROCEDURE
1.  Initialize variables: velocity factor V, c, $\gamma_{max}$, Coefficient c1, r1, MR flag, and no. of iterations
    /* Calculate Ranks of Workflows Tasks in DAG using HEFT Algorithm */
2.  Send workflows to HEFT
3.  For Each Task in DAG Do
4.    Calculate average execution time of all VMs
5.      If Task $t_i$ is the last Task Then
6.        Rank value of $t_i$ = its average execution time
7.      Else
8.        $rank_u(t_i) = WAvg_i + Max\ t_j \epsilon\ Succ\ (t_i) \left(CAvg_{ij} + rank_u(t_j)\right)$
            Where $WAvg_i$ is average execution cost
            Succ ($t_i$) is set of immediate successor of task $t_i$
            $CAvg_{i,j}$ is average communication cost
9.      End If
10. End For
11. Assign Tasks to VMs according to HEFT Rank
12. If Solution is not Optimized Then
13.   Generate a set of Cats by the Population generated by HEFT of Size S
14.   Initialize the velocity V of each Cat
15.     While No. of Iterations not Exceeded Do
16.       For K=1 to S
17.         According to Mixing Ratio (MR) flag Distribute Cats to Seeking and Tracing Modes
18.         If current $Cat_K$ is in Seeking Mode Then
19.           Generate S (SMP) Copies of $Cat_K$ and Spread them in D Dimensions where each Cat
              has a velocity ($V_{K,D}$)
20.           Evaluate the Fitness value of all Copies and Discover Best Cats ($X_{BEST,D}$)
21.           Replace Original $Cat_K$ with the Copy of Best Cats ($X_{BEST,D}$)
22.         Else If current $Cat_K$ is in Tracing Mode Then
23.           Compute and Update $Cat_K$ velocity by following equations:
24.             $\gamma = \gamma_{max} \times \exp\left(-c \times \left(\frac{itr}{itr_{max}}\right)^c\right)$
                Where, $\gamma$ is a weight factor calculated by Self-Motivated Inertia Weight method
                $\gamma_{max}$ and $c$ are constant factors greater than 1, both are set as 2.
25.             $V_{K,D} = \gamma \times V_{K,D} + \left(c1 \times r1 \times \left(X_{BEST,D} - X_{K,D}\right)\right)$
                Where, D = 1, 2, 3, …, $D_n$.
                c1 is acceleration coefficient, r1 is random number in the range of [0, 1]
26.           Update the position of every dimension of $Cat_K$ by using following equation:
27.             $X_{K,D} = X_{K,D} + V_{K,D}$
28.           Evaluate Fitness of all Cats and Discover Best Cats ($X_{BEST,D}$)
29.         End If
30.         Update Best Cats ($X_{BEST,D}$) in Memory
31.       End For
32.     End While
33. End If
34. return (Optimal Solution)
END PROCEDURE
```

Figure. 1 Pseudo-code of proposed algorithm H-CSO

Cat_K is replaced with that best cat in dimension D and saved into memory.

b) Tracing Mode: In tracing mode, the position and velocity of the current Cat_K is replaced by the given Eq. (3), (4), and (5).

$$\gamma = \gamma_{max} \times \exp\left(-c \times \left(\frac{itr}{itr_{max}}\right)^c\right) \quad (3)$$

Here, γ is the weight factor calculated by the above Eq. (3), γ_{max} and c are constant values and set to 2.0. itr is the current iteration number and itr_{max} is the maximum number of iterations.

This γ (SMIW factor) is integrated into the following Eq. (4) for balancing the velocity of the tracing mode.

$$V_{K,D} = \gamma \times V_{K,D} + \left(c1 \times r1 \times \left(X_{BEST,D} - X_{K,D}\right)\right) \quad (4)$$

Here, D = 1, 2, 3, …, D_n. c1 is the acceleration coefficient i.e. 1.5, r1 is set as a random number between [0, 1].

Now, with the help of Eq. (5) the position of the current Cat_K is updated.

$$X_{K,D} = X_{K,D} + V_{K,D} \quad (5)$$

Here, $X_{K,D}$ is the position, and $V_{K,D}$ is the velocity of cats in dimension D.

After this, the fitness of the Cats is checked and updated in the memory.

Step 5: This process continues till the stopping condition is not met as displayed in Fig. 1.

The pseudo-code of the newly designed algorithm is represented in Fig. 1. The proposed algorithm is able to overcome the initialization problems of the original CSO algorithm due to the HEFT policy. It speeds up the proposed algorithm for better convergence due to pre-processing of the workflows and the SMIW method overcomes the problem of escaping the Cats from the search space, which also improves the results.

5. Simulation setup

To measure the performance of introduced H-CSO and other existing scheduling algorithms, an open-source tool named CloudSim has been used which is written in Java. CloudSim provides an environment to simulate the scheduling, load balancing, and other policies. The experiments have been done over a machine with Intel (R) Core (TM) i3-5005U CPU @ 2.00 GHz, a RAM of 4.00 GB, 1 TB HDD, and running a Windows 10 64-bit operating system [28].

5.1 Parameters

For experiments, one PowerDatacenter has been created which has 25 GB RAM, 1000 MIPS speed per virtual machine, storage 1 TB, and bandwidth 50000 bps. The DC's OS is Linux. The system architecture of this DC is x86, VMM is Xen. Table 2 depicts the parameters configured for a simulation environment with heterogeneous VMs in the simulator. VMs contain 1 PE, 512-1024 MB RAM, 1000-1500 bps Bandwidth, 500-1000 MIPS of Processing Element and, 10000 MB Image size. The scheduling policy has been set as Time Shared. The required parameters of ACO, PSO, and CSO are also depicted in Table 2 [28].

Table 2. Simulation parameters

PowerDatacenter	
Parameter	**Values**
No. of Hosts	1
System Design	x86
VMM	Xen
Operating System	Linux
Number of Cloudlets	1000 in Each Workflow
Numbers of VMs	10, 20 and 30
CPU (PEs Number)	1
RAM per VM	512-1024 MB
Bandwidth	1000-1500 bps
Processing Elements Power per VM	500 – 1000 MIPS
Image Size	10000 MB
Policy Type	Time Shared
ACO	
No. of Initial Ants (m)	100
No. of Iterations	300
Q, Alpha, Beta, Gamma, Rho	1, 2, 1, 4, 0.05 Respectively
PSO	
No. of Particles	100
No. of Iterations	300
Local and Global Weights (C1 and C2)	1.5
CSO and H-CSO	
No. of Cats	100
Iterations	300
Weight Coefficient (C1)	1.5
r1 is Random Variable	[0, 1]
Mixing Ratio (MR) Percentage	Random [0-1] i.e. 0.2-0.3

5.2 Cost plan

The cost strategy of a system involves processing cost, memory usage cost, bandwidth cost, and storage cost as shown in Table 3.

5.3 Cloudlets

For experiments, scientific workflows datasets having different types of tasks have been used like CyberShake, Montage, Inspiral and, Sipht. Each dataset has 1000 tasks.

5.4 Performance metrics

Table 3. Cost Plan (in Indian Rupees)

Resource	Processor	RAM	Storage	Bandwidth
Size	500-1000 MIPS	512 MB	Unlimited	1000 bps
Cost	Rs. 3.0 per processor	Rs. 0.05 per MB	Rs. 0.1	Rs. 0.1 per MB

5.4.1 Makespan

Makespan [24] is defined as the finishing time of a group of tasks. It is calculated by the given Eq. (6).

$$Makespan = max\ (CT_i)_{ti \in tasks} \quad (6)$$

Here, CT_i is the completion time of Task T_i

5.4.2 Computing cost

Computing Cost is another vital metric because the end-users want a good service at a minimum cost. The cost can be calculated by Eq. (7).

$$Total\ Cost = \frac{MF+CF}{2} \quad (7)$$

Where, MF is a Movement Factor and CF is a Cost Factor

$$MF = \frac{1}{No.\ of\ Hosts/Datacenters}\left[\sum_{x=1}^{VMx}\left(\frac{No.\ of\ Migrations}{Used\ VM}\right)\right] \quad (8)$$

$$CF = \sum_{x=1}^{VMx}\left(\frac{Processing\ Cost \times Memory\ of\ Tasks}{VM \times Datacenter}\right) \quad (9)$$

Eq. (8) and Eq. (9) help to calculate the total cost as displayed in Eq. (7).

5.4.3 Fitness function

The fitness function has been used to check the best suitable VMs on the ground of various parameters like CPU utilization, Memory utilization, Makespan and Bandwidth utilization as shown in Eq. (10).

$$F_X = \frac{1}{Datacenter \times VMj}\left[\sum_{i=1}^{DCi}\sum_{j=1}^{VMj}\frac{1}{VM}\frac{CPU\ Utilized}{CPUij} + \frac{Memory\ Utilized}{Memoryij} + \frac{Makespan\ Utilized}{Makespanij} + \frac{Bandwidth\ Utilized}{Bandwidthij}\right] \quad (10)$$

6. Simulation results discussion

For experimentations, four scenarios with a group of 10, 20, and 30 VMs have been booked. For checking the performance of the introduced algorithm H-CSO, four other most widely used algorithms such as CSO, PSO, ACO, and Max-Min were used. The results which are obtained in terms of makespan are demonstrated in Table 4 with respect to all scenarios.

Fig. 2, 3, 4, and 5 are depicting the virtual machines at the x-axis and makespan at the y-axis.

Fig. 2 is depicting the results of makespan those have been found after simulation of the CyberShake_1000 workflow. This dataset is considered a data-intensive workflow created by Southern California Earthquake Center. It is used to analyze seismic hazards. It requires huge CPU power and memory utilization. It can be seen that the makespan is getting reduced with the increment of the number of virtual machines. This is because a large

Table 4. Makespan evaluation (in Sec)

Scenarios	VMs	Max-Min	ACO	PSO	CSO	H-CSO
Scenario - 1 CyberShake_1000	10	5839.75	5219.27	4707.43	4611.87	4513.29
	20	4590.90	4098.41	3778.52	3680.70	3494.52
	30	3793.57	2621.64	2430.22	2393.03	2358.73
Scenario – 2 Montage_1000	10	2711.35	2542.62	2340.24	2309.29	2269.37
	20	2459.23	2108.12	2080.54	2013.38	1977.27
	30	1972.58	1647.57	1329.35	1302.74	1217.25
Scenario – 3 Inspiral_1000	10	66309.76	57308.39	53073.25	52313.87	50948.55
	20	48301.02	43025.49	41217.13	40339.13	39239.24
	30	28411.54	26208.27	24387.89	23982.28	22537.97
Scenario – 4 Sipht_1000	10	39121.29	36507.45	34213.08	32813.83	31117.42
	20	30154.80	28309.74	25239.93	25209.13	23291.13
	30	26321.82	22401.98	20467.78	19119.29	18285.98

Received: June 26, 2021. Revised: July 20, 2021.

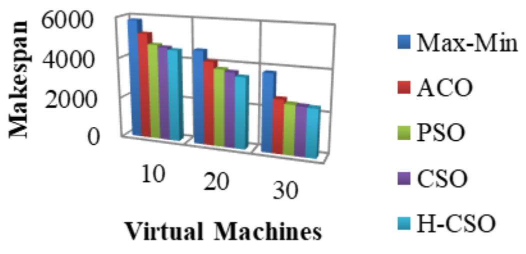

Figure. 2 Makespan evaluations for cybershake_1000 tasks

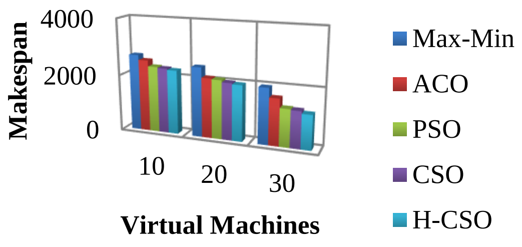

Figure. 3 Makespan evaluations for montage_1000 tasks

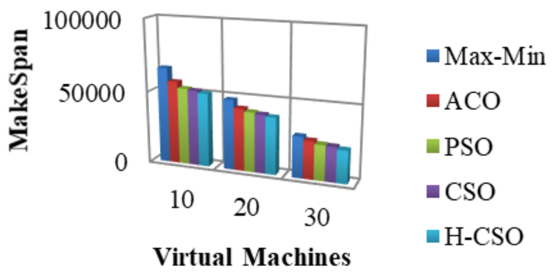

Figure. 4 Makespan evaluations for inspiral_1000 tasks

number of VMs work together for a particular problem without too much delay. With all sets of VMs i.e. 10, 20, and 30, H-CSO outperforms all other algorithms like Max-Min, ACO, PSO, and CSO. This is because H-CSO gives good convergence due to better initialization of VMs by the HEFT algorithm.

Fig. 3 is representing the results of makespan found after experiments with 1000 tasks in a workflow named Montage. The Montage workflow dataset is an astronomical application released and used by NASA as images for inputs. Most of this workflow has I/O-intensive data which requires less CPU power and memory. The graph is showing that the makespan is getting decremented while VMs get increased; this is because the same work is executed in less time with more than one virtual machine as compared to a single one. With 10, 20, and 30 VMs, H-CSO again performs better than all other algorithms. On the above-said dataset, H-CSO gave

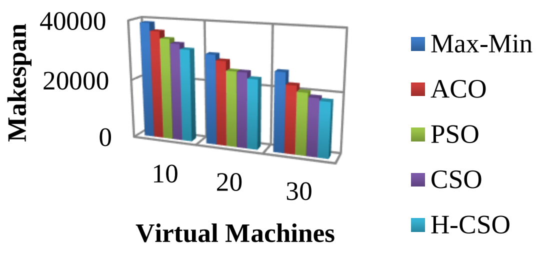

Figure. 5 Makespan evaluations for Sipht_1000 Tasks

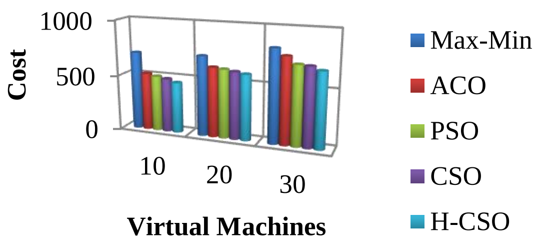

Figure. 6 Cost evaluations for cybershake_1000 tasks

better performance due to good global searching and improved tracing mode.

Fig. 4 is demonstrating the results of Makespan associated with the Inspiral_1000 dataset. This workflow comes from the field of physics and is used to analyse gravitational wave-related data. This dataset is CPU-intensive and requires a huge memory. Here, H-CSO performs better on 10, 20, and 30 VMs as compared to Max-Min, ACO, PSO, and CSO. Due to better convergence and effective tracing mode as compared to CSO, H-CSO performed well over CSO and other algorithms. The reason of the better convergence is the HEFT policy.

Fig. 5 is showing the makespan comparison for 1000 tasks belonging to Sipht workflow which are dependent in nature. This workflow is derived from the Harvard International Bioinformatics Center's project that represents an automated search for sRNA-encoding genes of various bacteria. This workflow requires high computational power, memory and, I/O devices. In this scenario, H-CSO again outperforms other algorithms for 10, 20, and 30 virtual machines due to the SMIW factor.

With these results, it is summarized that H-CSO is having a healthier makespan for all scenarios in contrast to other algorithms used in this simulation. From Table 4, Fig. 2, 3, 4, and 5, it can be noticed that the CSO and H-CSO work well. In overall scenarios, H-CSO works better than other existing algorithms because of better pre-processing of tasks and better global searching, and good convergence than others.

Received: June 26, 2021. Revised: July 20, 2021.

Table 5. Cost consumption evaluation (in Indian Rupees)

Scenarios	VMs	Max-Min	ACO	PSO	CSO	H-CSO
Scenario - 1 CyberShake_1000	10	704.08	517.27	499.24	484.20	457.57
	20	718.12	625.21	615.85	602.17	587.67
	30	830.50	770.53	707.23	701.19	670.23
Scenario – 2 Montage_1000	10	238.48	201.43	189.53	181.29	165.38
	20	241.19	219.07	202.63	190.52	183.58
	30	249.04	230.71	208.15	197.58	191.29
Scenario – 3 Inspiral_1000	10	4281.40	3831.43	3755.38	3661.28	3629.47
	20	4380.88	4024.85	3859.39	3753.38	3684.24
	30	5994.97	5741.13	5649.67	5429.13	5219.65
Scenario – 4 Sipht_1000	10	3154.92	2915.03	2859.33	2810.53	2753.07
	20	3620.48	3257.99	3198.45	2973.13	2915.29
	30	3907.58	3519.39	3460.17	3351.29	3107.40

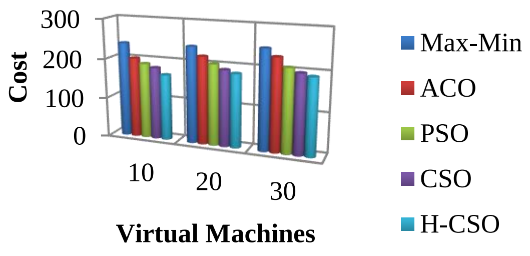

Figure. 7 Cost evaluations for montage_1000 tasks

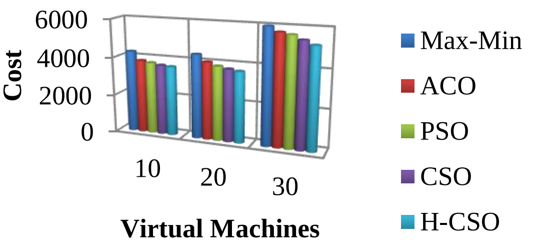

Figure. 8 Cost evaluations for inspiral_1000 Tasks

Table 5 is presenting the comparison of the cost of resources used in the simulation for all scenarios as per the cost plan given in Table 3 and Eq. (7). Fig. 6, 7, 8, and 9 are showing the virtual machines and cost at the x-axis and y-axis respectively.

Fig. 6 is depicting the cost comparison of resources used for CyberShake workflows tasks. The picture is showing that the proposed H-CSO is consuming lesser costs than other algorithms with all sets of VMs. The better results are achieved by H-CSO due to efficient workflow task migration among VMs as the SMIW method balances the tracing mode.

Fig. 7 is interpreting the cost comparison of resources used for 1000 tasks belonging to the Montage dataset.

While working on 10, 20 and, 30 VMs the cost of the CSO is having lesser better as compared to the PSO, ACO, and Max-Min algorithms. H-CSO

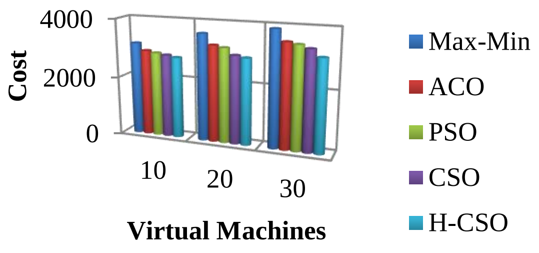

Figure. 9 Cost evaluations for sipht_1000 tasks

outperforms all algorithms over here just because of finding the best VMs without wasting too much time due to the SMIW integration in tracing mode.

Fig. 8 is demonstrating the cost comparison of resources used for 1000 Inspiral workflow tasks. When working on 10, 20, and 30 VMs; the cost is found minimum by using H-CSO as compared to other algorithms such as CSO, PSO, ACO, and Max-Min due to searching for suitable VMs as early as possible due to the HEFT and SMIW methods.

Fig. 9 is showing the cost comparison of resources used for the fourth scenario that is having 1000 dependent tasks belonging to the Sipht dataset. The graph tells that when working with 10, 20, and 30 virtual machines, the H-CSO is again working better in terms of the processing cost as compared to other algorithms shown in Fig. 9. H-CSO utilized lesser cost as compared to others because it is intelligent enough to identify and manage the best virtual machines and put the tasks on them at a suitable time due to the SMIW method.

With these results, it can be stated that the H-CSO algorithm has beaten CSO, PSO, ACO, and Max-Min algorithms with consuming lesser costs and having better makespan with overall levels in all scenarios due to better convergence and global optimization rate. It is just because of the HEFT policy, velocity management factor γ (SMIW). The CSO algorithm is winning the race at the second position because of having the inspirational properties of ACO and PSO and better convergence speed due to better global searching property as compared to the Max-Min, ACO and PSO.

7. Conclusion and future scope

IT industries and other organizations are moving on the cloud regularly; this causes load increment on the cloud servers. The workflow or task scheduling is required to get managed this load. Scientists have developed several algorithms in the field of workflow task scheduling in order to schedule the tasks on virtual machines. Many existing algorithms have numerous drawbacks. In this research paper, a new algorithm is introduced and simulated named H-CSO. It removes a few drawbacks of the CSO algorithm which is superior to other algorithms as found in the related work. The proposed H-CSO algorithm has been compared with the most widely used task scheduling algorithms such as CSO, PSO, ACO, and Max-Min with various datasets as described earlier. A total of four scenarios had been designed with a set of 10, 20, and 30 heterogeneous virtual machines in the CloudSim tool which is open source and written in Java.

For all scenarios, the proposed H-CSO algorithm outperformed for makespan while consuming less cost than other algorithms. This is because the HEFT algorithm pre-processes the workflows and puts them on the VMs in an efficient manner. The H-CSO is initialized with a population generated by the HEFT algorithm despite random initialization. The SMIW method balances the velocity calculation method of the tracing mode of the CSO. It protects the Cats to go outside the search space. The proposed algorithm is a generalized one and works better with all kinds of scientific datasets i.e. workflows. The efficiency of the proposed algorithm H-CSO found 2.99%, 2.87%, 3.35% and 5.77% better for CyberShake_1000, Montage_1000, Inspiral_1000, and Sipht_1000 datasets respectively as compared to the CSO algorithm in respect to makespan reduction. For cost reduction, the proposed algorithm H-CSO is found 4.03%, 5.12%, 2.42% and 3.93% effective for the CyberShake_1000, Montage_1000, Inspiral_1000, and Sipht_1000 scientific datasets correspondingly as compared to the standard CSO. This research showed that cloud customers can execute their services in faster mode by paying a little extra cost.

In the end, it is determined that the introduced H-CSO algorithm is found effective than all other algorithms in the account of makespan with the minimum consuming cost. Still, there are chances of the enhancement in H-CSO for restricting it from getting stuck in local minima and premature convergence. This problem may occur when the number of iterations is immense in number.

In the future, it can be enhanced by integrating any suitable local searching technique or any other method for a better convergence of the proposed algorithm.

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

The paper drafting, editing, and simulation have been carried out by the 1st author. The review, directions for simulation, and supervision process was carried out by the 2nd author.

References

[1] 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. 9, No. 1, pp. 33-55, 2018.

[2] N. Zanoon and D. Rawshdeh, "STASR: A New Task Scheduling Algorithm for Cloud Environment", *Network Protocols and Algorithms*, Vol. 7, No. 2, pp. 81-95.

[3] J. K. Konjaang, J. Y. Maipan-uku, and K. K. Kubuga, "An Efficient Max-Min Resource Allocator and Task Scheduling Algorithm in Cloud Computing Environment", *International Journal of Computer Applications*, Vol. 142, No. 8, pp. 25-30, 2016.

[4] A. A. maamari and F. A. Omara, "Task Scheduling using PSO Algorithm in Cloud Computing Environments", *International Journal of Grid Distribution Computing*, Vol. 8, No. 5, pp. 245-256, 2015.

[5] I. R. K. Raju, P. S. Varma, M. V. R. Sundari, and G. J. Moses, "Deadline Aware Two Stage Scheduling Algorithm in Cloud Computing", *Indian Journal of Science and Technology*, Vol. 9, No. 4, pp. 1-10, 2016.

[6] R. J. Priyadarsini and L. Arockiam, "Performance Evaluation of Task Scheduling in Cloud Environment using Soft Computing Algorithms", *International Journal of Computer Science and Network*, Vol. 4, No. 2, 387-391, 2015.

[7] B. Santosh and D. H. Manjaiah, "A Hybrid AvgTask-Min and Max-Min Algorithm for Scheduling Tasks in Cloud Computing", In:

Proc. of IEEE International Conference on Control, Instrumentation, Communication and Computational Technologies, Kumaracoil, India, pp. 325-328, 2015.

[8] R. Kapur, "A Workload Balanced Approach for Resource Scheduling in Cloud Computing", In: *Proc. of IEEE 8th Internation Conference on Contemporary Computing (IC3)*, Noida, India, 2015.

[9] S. T. Dehkordi and V. K. Bardsiri, "TASA: A New Task Scheduling Algorithm in Cloud Computing", *Journal of Advances in Computer Engineering and Technology*, Vol. 1, No. 4, pp. 25-32, 2015.

[10] J. Bhagwan and S. Kumar, "An Intense Review of Task Scheduling Algorithms in Cloud Computing", *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 5, No. 11, pp. 605-611, 2016.

[11] K. Etminani, M. Naghibzadeh, and N. R. Yanehsari, "A Hybrid Min-Min Max-Min Algorithm with Improved Performance", *Department of Computer Engineering, Ferdowsi University of Mashad*, Iran.

[12] 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.

[13] R. K. Jena, "Multi Objective Task Scheduling in Cloud Environment using Nested PSO Framework", In: *Proc. of Procedia Computer Science, Elsevier*, Vo. 57, pp. 1219-1227, 2015.

[14] A. M. Ahmed, T. A. Rashid, and A. M. Saeed, "Cat Swarm Optimization Algorithm: A Survey and Performance Evaluation", *Computational Intelligence and Neuroscience*, Vol. 2020, pp. 1-20, 2020.

[15] S. Yassir, Z. Mostapha, and T. Cluade, "E-HEFT: Enhancement Hetrogeneous Earliest Finish Time algorithm for Task Scheduling based on Load Balancing in Cloud Computing", In: *Prof. of International Conference on High Performance Computing& Simulation*, Orleans, France, pp. 601-609, 2018.

[16] P. Pradhan, P. K. Behera, and B. N. B. Ray, "Enhanced Max-Min Algorithm for Resource Allocation in Cloud Computing", *International Journal of Advanced Science and Technology*, Vol. 29, No. 8, pp. 1619-1628, 2020.

[17] S. Devipriya and C. Ramesh, "Improved Max-Min Heuristic Model for Task Scheduling in Cloud", In: *Proc. of International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, Chennai, India, pp. 883-888, 2013.

[18] X. Wei, "Task Scheduling Optimization Strategy Using Improved Ant Colony Optimization Algorithm in Cloud Computing", *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[19] G. R. N. Reddy and S. Phanikumar, "Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud Computing", *International Journal of Intelligent Engineering & Systems*, Vol. 11, No. 3, pp. 242-250, 2018.

[20] A. M. S. Kumar and M. Venkatesan, "Multi-Objective Task Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment", *Wireless Personal Communications*, Vol. 107, pp. 1835-1848, 2019.

[21] Q. Guo, "Task Scheduling Based on Ant Colony Optimization in Cloud Environment", In: *Proc. of AIP Conference Proceedings*, Vol. 1834, 2017.

[22] Y. Moon, H. Yu, J. Gil, and J. Lim, "A Slave Ants Based Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing Environments", *Human-centric Computing and Information Sciences*, Vol. 7, No. 28, 2017.

[23] H. Yu, "Evaluation of Cloud Computing Resource Scheduling Based on Improved Optimization Algorithm", *Complex & Intelligent Systems*, 2020.

[24] H. Saleh, H. Nashaat, W. Saber, and H. M. Harb, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment", *IEEE Access*, Vol. 7, 2019.

[25] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, A. Abraham, and N. M. Dankolo, "Cloud Customers Service Selection Scheme Based on Improved Conventional Cat Swarm Optimization", *Neural Computing and Applications*, Vol. 32, pp. 14817-14838, 2020.

[26] S. Bilgaiyan, S. Sagnika, and M. Das, "Workflow Scheduling in Cloud Computing Environment Using Cat Swarm Optimization", In: *Proc. of IEEE International Advance Computing Conference (IACC)*, pp. 680-685, 2014.

[27] D. Gabi, A. S. Ismail, and N. M. Dankolo, "Minimized Makespan Based Improved Cat Swarm Optimization for Efficient Task Scheduling in Cloud Datacenter", In: *Proc. of 3rd High Performance Computing and Cluster Technologies Conference*, Guangzhou, China, pp. 16-20, 2019.

[28] J. Bhagwan and S. Kumar, "Performance Evaluation of Meta-Heuristic Algorithms for Task Scheduling in Cloud Environment",

International Journal of Advanced Research in Science, Communication and Technology, Vol. 5, No. 2, pp. 87-93, 2021.