An HC-CSO Algorithm for Workflow Scheduling in Heterogeneous Cloud Computing System

Jai Bhagwan[1], Sanjeev Kumar[2]
Department of Computer Science & Engineering
Guru Jambheshwar University of Science & Technology
Hisar, India

***Abstract*—Many scientists are using meta-heuristic techniques for dynamic workflow task scheduling in the area of cloud computing systems to get optimum solutions. Many swarm intelligent algorithms have been designed so far which are having many limitations as some get trapped in local optima, a few are having low convergence speed, some are having poor global search facilities, etc. Still, there is a requirement of designing a new algorithm or modification of existing algorithms to overcome the limitations of the existing techniques. A new Hybrid Cat Swarm Optimization algorithm named H-CSO was designed inspired by the HEFT algorithm and the initialization problem of the Cat Swarm Optimization was overcome. Still, that algorithm has a limitation of getting stuck in local minima. To overcome this algorithm a part of the Crow Search Algorithm has been integrated into H-CSO and described in this paper. After simulation, it was found that the new hybrid algorithm named HC-CSO outperforms CSO and H-CSO.**

Keywords*—Cloud computing; Crow Search Algorithm (CSA); Cat Swarm Optimization (CSO); H-CSO; HC-CSO; HEFT; Self-Motivated Inertia Weight (SMIW); Virtual Machines (VMs)

I. INTRODUCTION

Information Technology has been reshaped by the evolution of cloud computing technology via big storage facilities, high-performance computing, and other hardware and software services. The current technology includes the evolution of computing eras where computers were connected via the internet that took the form of distributed computing. This further transformed into cluster computing, cluster to grid computing, and then cloud computing [1]. The major aim of cloud computing technology is to provide high-performance computing services at the minimum cost. The cloud technology shifted the users' data from client machines to network-abled machines which are having high-powered processors and hardware parts. Cloud computing provides services in the form of Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [15]. Any end-user can pick up the services as per his requirements. The main benefit of the cloud is that it doesn't include geographical boundaries to provide the services to the end-users [2]. This means that the users need not to know the physical locations of the service providers and computing datacenters. Cloud technology is flexible because a user can increase the number of services and drop whenever it is required due to the pay-per-usage policy. Various cloud service providers are Amazon Web Services, Microsoft, Google, Rackspace, salesfornce.com, etc. The cloud system can be classified into four categories such as private cloud, public cloud, community cloud, and hybrid cloud [3].

The performance of the cloud can be improved at various levels such as network level, scheduling level, database level, etc. After studying various research papers, it can be seen that numerous algorithms have been designed for workflow task scheduling, load balancing, energy consumption management, etc. in cloud computing. Workflow task scheduling is one of the major areas where a lot of improvement is required. Task scheduling can be of two types static and dynamic scheduling. In static scheduling, the execution times of the tasks are pre-estimated or known but in the case of dynamic scheduling, it is not known. This era is of dynamic task scheduling. So, dynamic task scheduling algorithms are required to be improved. The optimization techniques play an important role to improve the cloud scheduling problem nowadays. A few famous techniques are Ant Colony Optimization, Particle Swarm Optimization, and Cat Swarm Optimization, etc. Cat Swarm Optimization belongs to Swarm Intelligent (SI) family [4]. The algorithms used in this paper are as under:

1) Cat swarm optimization: This is an intelligent algorithm originally developed in the year of 2006. This algorithm is inspired by the behaviour of the original cat which is having to two modes known as seeking (resting mode) and tracing (attacking mode). Here, N numbers of cats are generated randomly and each cat denotes a solution, its position, a flag, and fitness value. M dimensions in the search space represent the position, and each dimension in the search space is having its self-velocity. The flag is used to identify that the cat is in either seeking mode or in tracing mode and this flag is set by a parameter known as Mixing Ratio (MR). After finding the fitness, the best cat is stored in the memory at each iteration and finally, the best solution or cat is identified at the end of the final iteration. The two modes of Cat Swarm Optimization are described below [1][21]:

i) Seeking mode: This mode represents the resting mode of a cat and four parameters play an important role in it: SMP (Seeking Memory Pool), SRD (Seeking range of the Selected Dimension), CDC (Counts of Dimension to change), and SPC (Self Position Consideration). In SMP, one position is selected by a cat randomly for moving to the next position. Let's say, the SMP is set to 10, now for every cat 10 new random positions will be generated and one among them will be

selected randomly for movement. SRD and CDC will decide the randomization of the new positions. How many dimensions need to be mutated that is decided by the CDC factor which is in the interval between [0 to 1]. The amount of mutation is defined by SRD for the dimensions selected by CDC. The Boolean value SPC will consider the candidate cats for the next iteration from the current position. Let's assume, if SPC is true then for each cat SMP-1 candidates will be generated instead of SMP because the current position will be decided from them. Seeking Mode steps are given as:

a) Generate S Copies of Seeking Cat_K equal to the SMP value.

b) For each copy, change at a random dimension of Cats as per CDC by applying SRD operator as:

$$Cat\ nD_{new} = (1 + rand * SRD) * Cat\ nD_{old} \quad (1)$$

Where, $Cat\ nD_{old}$ is the current position and $Cat\ nD_{new}$ is the next new position, n is the numbers of Cats and D is the dimension, rand is a random variable between [0, 1] interval.

c) Evaluate fitness of all candidate/ changed Cats and find Best Cats ($Cat_{Best,D}$).

d) Replace the position of S Cats by picking up the Best Cats randomly.

ii) Tracing Mode: This mode follows the tracing behaviour of real cats.

a) At first, velocity values are computed and assigned to all the dimensions of a Cat_K position by the following equation 2.

$$V_{K,D} = V_{K,D} + \left(c1 * r1 * \left(X_{Best,D} - X_{K,D}\right)\right) \quad (2)$$

Where, $V_{K,D}$ is the velocity, c1 is an acceleration coefficient, r1 is a random variable between [0, 1], $X_{Best,D}$ is the best cat position, $X_{K,D}$ is the current cat position and D is the dimension.

b) If velocity is going beyond the upper range set it within the range.

c) Update the position of the Cat_K by using the following equation 3.

$$X_{K,D} = X_{K,D} + V_{K,D} \quad (3)$$

2) Crow search algorithm: Crow family is treated as one of the most intelligent bird groups. Their brain is considered slightly lower than a human being, based on their body-to-brain ratio. Crow is a very well famous thief as to watches other bird's food and steals it after the victim bird leaves its place of food. This intelligent behaviour of the crow can be used to solve and optimize real-world problems. The CS Algorithm can be described with the help of a pseudo-code in Fig. 1 [22].

Crow Search Algorithm

1. Randomly initialize the positions of a group of N crows, fl (flight length), r_i, r_j and max. iterations
2. Evaluate the position of all the crows
3. Initialize the memory of every crow in the group
4. **While** itr < itr_{max}
5. **For** i = 1 to N
6. Randomly choose one of the crows to follow, let it be Crowj
7. Define the awareness probability
8. **If** $r_j \geq AP_{i,itr}$ **Then**
9. $x_{i,itr} = x_{i,itr} + r_i * fl_{i,itr} * \left(m_{j,itr} - x_{i,itr}\right)$ (4)
10. **Else**
11. $x_{i,itr}$ = a random position of a crow in search space
12. **End If**
13. **End For**
14. Check the fitness of new positions
15. Evaluate the new position of the crows
16. Update the memory of crows
17. **End while**

Fig. 1. Pseudo-Code of Crow Search Algorithm.

First of all, the positions of all crows in a group and other parameters have been initialized. N is the number of crows, r_i and r_j are random variables between 0 to 1. After evaluation of the positions of all crows, the memory of each crow is initialized. As described earlier, the behavior of the crow is to chase a bird to steal its food. So, a random crow from the search space is selected say it be a $Crow_j$. $Crow_i$ will follow it and will steal its food whenever the $Crow_j$ will leave its place after hiding its food. If the random variable r_j will be greater than awareness probability the position of the current $Crow_i$ will be updated by using equation 4 otherwise the position will be updated randomly from the search space. Flight Length (fl) of the crow will decide the local or global search. If the fl is less than 1 then it will work for local search otherwise the global search will take place. Then the new positions or solutions will be stored in the memory and this process will be continued till the termination condition will not be satisfied [22].

In this paper, the limitation of getting stuck in the local minima of the H-CSO algorithm has been improved by integrating the local search part of the CSA. The details are given in the coming sections.

The rest of the paper is structured as follows: Section II covers the related work. In Section III, the proposed methodology is described. Section IV is having a description of the simulation setup and simulation results and discussion are covered in Section V. Section VI is summarizing the conclusion and future scope. In the end, the research papers' references are given.

II. Related Work

Existing work in the area of workflow and task scheduling can be found in the following literature review: In [1] the scientists proposed multi-objective Cat Swarm Optimization based on the Simulated Annealing technique. The Simulated Annealing (SA) technique is incorporated in the local search of the proposed algorithm and the SA is enhanced by the Orthogonal Taguchi approach. The parameters like execution time and execution cost are considered for performance

measurement. The proposed technique worked better as compared to Multi-Objective Ant Colony Optimization, Multi-objective Genetic Algorithm, and Multi-Objective Particle Swarm Optimization. The authors [2] proposed a Multi-Objective Cat Swarm Optimization algorithm and after comparison with the existing Multi-Objective PSO technique, the proposed approach was found better in the account of energy consumption, execution cost, and execution time. The researchers in [3] introduced a Cat Swarm Optimization-based technique for workflow scheduling. The results were compared with PSO (Particle Swarm Optimization) and found that the CSO reduces the processing cost, made a good load balance, and gave the optimum results in less iteration. In [5] a Hybrid Particle Swarm Optimization algorithm using the Hill-Climbing technique was proposed by the scientists. The proposed algorithm was found effective in terms of makespan after experiments. The scientists [6] introduced a Binary Hybrid Particle Swarm Optimization and Gravitational Search algorithm for load balancing of virtual machines (VMs). The experimental results showed that the proposed algorithms worked better than the existing Binary PSO load balancing algorithm in terms of load balancing. In [7] the researchers offered a new HPSOGWO algorithm that is a combination of Particle Swarm Optimization and Grey Wolf Optimization. The idea behind this algorithm was to improve the exploitation of PSO and exploration of GWO for making the better strength of the proposed algorithm. The results concluded that the HPSOGWO is better than standard PSO and GWO variants concerning solution stability, convergence speed, and quality. The authors in [8] proposed an IPSO (Improved Particle Swarm Optimization) algorithm to improve the allocation of large-length tasks. The introduced algorithm outperformed despite Ant Colony, Honey Bee, and Round-Robin algorithm in accounts of load balance, makespan, and degree of imbalance. In [9] authors presented a Particle Swarm Optimization based technique for workflow scheduling and found that the Particle Swarm Optimization based technique saved the cost equal to 3 times as compared to BRS (Best Resource Selection) algorithm. Also, the presented technique balanced the load efficiently. The authors of the paper [10] proposed a PSO (Particle Swarm Optimization) based scheduling technique for independent tasks. The proposed technique was improved using a load balancing strategy. The newly introduced method was compared with Improved PSO, Round-Robin, and existing load balancing techniques. The experimental results showed that the proposed technique is better than the above said three algorithms for resource utilization and makespan. The scientist [11] introduced the MPSO (Modified Particle Swarm Optimization) algorithm for the reduction of cost as compared to the existing Particle Swarm Optimization algorithm. The simulation results showed that the proposed technique worked better. In paper [12] the authors modified Particle Swarm Optimization by modifying the parameters like MIPS and Bandwidth for effective load balancing. The simulation results showed that the proposed model resulted in a reduction of execution time. The researchers of the paper [13] compared a task scheduling strategy based on Ant Colony Optimization with FCFS and RR (Round-Robin). The simulation results proved that the ACO outperforms traditional FCFS and Round-Robin algorithms. The authors [14] developed a hybrid optimization technique using Flower Pollination and Grey Wolf Optimization. The PEFT algorithm was used for the initialization of the proposed method for workflows. The simulation resulted that the proposed method is effective as compared to Flower Pollination with Genetic Algorithm in terms of cost and reliability. In [16] the researchers designed an Improved Social Learning Optimization Algorithm by introducing the Small Vector Position method for task scheduling problems. After simulation, it was found that the proposed approach worked well as compared to GA and PSO-based techniques. In the paper [17], the researchers proposed an Adaptive Cost-based Task Scheduling technique to scheduling the tasks between virtual machines at minimum cost. The simulation results concluded that the proposed technique is performing well in terms of communication cost, execution time, CPU utilization and execution cost rather than cost-efficient task scheduling. In [18] research paper, the scientists implemented a Dynamic Adaptive Particle Swarm Optimization algorithm to enhance the efficiency of Particle Swarm Optimization for better makespan, resource utilization. The authors also proposed an algorithm named MDAPSO. DAPSO and MDPSO worked better than the original Particle Swarm Optimization. The authors of [19] research paper proposed an Improved Particle Swarm Optimization based technique to solve workflow scheduling problems in cloud systems. The proposed method worked better as compared to existing state-of-art methods. In [20], the scientists proposed a cloud scheduling model named Task Scheduling System. The proposed Genetic Algorithm-Chaos Ant Colony Optimization worked better than ACO and GA algorithms in respect to cost and convergence speed.

From the above study, it is found that the ACO, PSO, and CSO algorithms are optimizing the scheduling of independent as well as workflows tasks. The ACO algorithm is performing well for local searching; the PSO and CSO algorithms are good for global searching and get stuck in local optima easily. Various researchers integrated a few techniques and formulae in these algorithms to improve the performance of the cloud. But, these techniques are old. The CSO algorithm is a good performer among the ACO and PSO; the H-CSO is better than the CSO.

So, a new method is required to integrate into the H-CSO so that its limitation of getting trapped in local optima can be overcome. Hence, the local searching part of the Crow Search Algorithm has been integrated into the H-CSO algorithm. The details are indicated in the proposed methodology section.

III. Proposed Methodology

From the related work, it is learned that a single algorithm could not be able to give better results for workflow task scheduling.

Proposed Algorithm

Input (Tasks (T_1, T_2, T_3 … T_n), Virtual Machines (VM_1, VM_2, VM_3 … VM_m)
Output (Optimal Makespan and Cost of n Tasks on m VMs)

BEGIN PROCEDURE

1. Initialize fl (flight length), r_K, velocity factor V_K, c, γ_{max}, Coefficient c1, MR flag, and no. of iterations
   /* Calculate Rank of Workflows Tasks in DAG using HEFT Algorithm */
2. Feed workflows in 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. $rank_u(t_i) = WAvg_i + Max\ t_j \in succ(t_i)\ (CAvg_{ij} + rank_u(t_j))$
   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 not Optimized **Then**
13. Generate a set of Crows by the Population generated by HEFT of Size N
14. **While** No. of Iterations not Exceeded **Do**
15. **For** K=1 to N
16. Update the positions by the following equation : // Do local search
17. $X_{K,D} = X_{K,D} + r_K * fl_{K,D} * (M_{L,D} - X_{K,D})$
    Where, $X_{K,D}$ is current position of $Crow_K$, r_K is uniformly distributed random number [0, 1]
    $fl_{K,D}$ is flight length (less than 1 i.e. 0.5) of the $Crow_K$ at current iteration
    $M_{L,D}$ is present best location of $Crow_K$ in Dimension D
18. Feed the population generated by Local CSA in H-CSO // Do local and global search as given below
19. Assign the velocity V_K to each Cat
20. According to Mixing Ratio (MR) flag Distribute Cats to Seeking and Tracing Modes
21. **If** current Cat_K is in Seeking Mode **Then**
22. Generate S (SMP) Copies of Cat_K and Spread them in D Dimensions where each Cat has a velocity ($V_{K,D}$)
23. Evaluate the Fitness value of all Copies and Discover Best Cats ($X_{BEST,D}$)
24. Replace Original Cat_K with the Copy of Best Cats ($X_{BEST,D}$)
25. **Else If** current Cat_K is in Tracing Mode **Then**
26. Compute and Update Cat_K velocity by following equations:
27. $\gamma = \gamma_{max} * \exp\left(-c * \left(\frac{itr}{itr_{max}}\right)^c\right)$
    Where, γ is a weight factor calculated by Self-Motivated Inertia Weight method
    γ_{max} and c are constant factors greater than 1, both are set as 2.
28. $V_{K,D} = \gamma * V_{K,D} + \left(c1 * r1 * (X_{BEST,D} - X_{K,D})\right)$
    Where, D = 1, 2, 3, …, M.
    c1 is acceleration coefficient, r1 is random number in the range of [0, 1]
29. Update the position of every dimension of Cat_K by using following equation:
30. $X_{K,D} = X_{K,D} + V_{K,D}$
31. Evaluate Fitness of all Cats and find out Best Cats ($X_{BEST,D}$) with Best Fitness
32. **End If**
33. Update Best Cats ($X_{BEST,D}$) in Memory
34. **End For**
35. **End While**
36. **End If**
37. return (Optimal Solution)

END PROCEDURE

Fig. 2. Pseudo-Code of Proposed HC-CSO Algorithm.

Many scientists advised improving the existing algorithms. The H-CSO algorithm is a combination of the HEFT, Cat Swarm Optimization algorithm, and Self-Motivated Inertia Weight useful in overcoming the velocity outrange problem of the standard CSO. As said earlier, this H-CSO algorithm gets trapped in local minima in the case of a large search space and complex workflows' tasks environment. A few limitations of the H-CSO algorithm are described below in short:

1) The H-CSO algorithm is having only a good global searching capacity.

2) The H-CSO algorithm gets trapped in local minima due to a large number of cats is always residing in seeking Mode as compared to tracing mode.

3) Due to a lack of balance between seeking and tracing modes, optimal results could not be got using the H-CSO algorithm.

In the proposed algorithm named HC-CSO, a local search part of a well-known Crow Search Algorithm is integrated into the H-CSO for avoiding it getting trapped in the local optima. The working of the HC-CSO algorithm is described as:

First of all, the initialization of various parameters takes place then pre-processing of workflows tasks is executed with the HEFT method as shown in the pseudo-code of the proposed algorithm. After the pre-processing of workflows, the tasks are assigned to available VMs. If this solution is getting optimized at the very first stage then the algorithm is stopped and returns the optimized schedule otherwise the initial solution generated by the HEFT algorithm is fed to the CSA algorithm. Here, a number of N Crows are generated using the population obtained by the local Crow Search Algorithm. Then, the population got from the local CSA is fed to the H-CSO algorithm. Now, N numbers of Cats are generated and velocity value V_K is assigned to each cat for further processing. In the next phase, the cats are randomly distributed into the seeking and tracing mode as per the Mixing Ratio (MR) rate and flags are set to each cat. If the current Cat_K is found in the seeking mode as per flag value, this mode gets executed otherwise the tracing mode gets executed. The position and velocities of the cats in the tracing mode are updated using the equations given in the pseudo-code in Fig. 2 and the best cats get stored in the memory in the form of solution. This process is continued until the termination condition is not matched and finally an optimal solution is returned in the form of better makespan and cost. The proposed algorithm has both the best local and global searching capacity, so the results could be reached at an optimum level. The proposed algorithm is also useful to avoid the premature convergence of the H-CSO method. The pseudo-code of the proposed algorithm named HC-CSO has been described step by step in Fig. 2.

IV. Simulation Setup

The simulation environment has been created in the CloudSim tool for simulating the different workflows used in this paper. Various experiments were carried out over a computing machine having the configuration as Processor – Intel ® Core ™ i3-5005U at a speed of 2.0 GHz, RAM – 4.0 GB, HDD – 1 TB, and OS – Windows 10.

A. Parameters

For simulation, a PowerDatacenter has been designed having the configuration as RAM – 25 GB, MIPS per VM – 1000 MIPS, Storage – 1 TB, Bandwidth – 50000 bps. Rest configurations of the heterogeneous cloud environment are depicted in Table I. The scheduling policy was set as Time Shared. The parameters of the CSO, the H-CSO and, the proposed HC-CSO are also summarized in Table I.

TABLE I. Simulation Parameters

PowerDatacenter	
Parameters	**Values**
Number of Hosts	1
System Architecture	x86
VMM	Xen
OS	Linux
Number of Cloudlets Cloudlets Length Type	1000 in Each Workflow Random
Numbers of VMs	10, 20 and 30
CPU (PEs Number)	1
RAM per VM	512-1024 MB
Bandwidth	1000-1500 bps
Processing Elements per VM	500 – 1000 MIPS
Image Size	10000 MB
Policy Type	Time Shared
Standard CSO and H-CSO	
No. of Cats	100
Iterations	300
Weights (C1)	1.5
r1and r_k (Random Variables)	[0, 1]
Mixed Ratio Percentage	Random Range [0, 1] i.e. 0.2-0.3
HC-CSO (Proposed Algorithm)	
No. of Cats	100
Iterations	300
Weights (C1)	1.5
r1(Random Variable)	[0, 1]
Mixed Ratio Percentage	Random Range [0, 1]
fl (Flight Length)	0.5

B. Cost Plan

The cost plan (in Indian Rupees) of workflow scheduling is depicted in Table II.

TABLE II. Cost Plan

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

C. Cloudlets

Cloudlets are called tasks to be submitted for execution on virtual machines. In this paper, the scientific workflows named CyberShake, Montage, Inspiral, and Sipht have been used to test the proposed algorithm along with others. Each workflow is having 1000 tasks.

D. Performance Metrics

There are several performance metrics like Makespan, Processing Cost, Waiting Time, Response Time, Energy Consumptions, and Resources Utilization, etc. to test the performance of the algorithms. In this paper, makespan and cost are used to measure the performance of the proposed algorithm, and these parameters are given in the coming topics.

1) Makespan: Makespan [23] is referred to the maximum time taken for finishing the last task in a group. It is the most widely used metric to measure the performance of a scheduling algorithm. A lesser makespan decides that the algorithm is efficient enough. The makespan is computed by the given equation 5.

$$\text{Makespan} = \max (CT_i)_{\ ti \in tasks} \quad (5)$$

Where, CT_i is the completion time of $task_i$

2) Processing cost: Processing Cost is along with makespan is another important performance metric because cloud service providers want to give efficient services at the minimum costs in this competitive environment. The processing cost can be measured by equation 6.

$$Total\ Cost = \frac{MF+CF}{2} \quad (6)$$

Where, MF is Movement Factor

CF is Cost Factor

$$MF = \frac{1}{No.\ of\ Hosts/Datacenters}\left[\sum_{x=1}^{VMx}\left(\frac{Number\ of\ Migrations}{Used\ VM}\right)\right] \quad (7)$$

$$CF = \sum_{x=1}^{VMx}\left(\frac{Processing\ Cost * Memory\ of\ Tasks}{VM * Datacenter}\right) \quad (8)$$

3) Fitness function: Equation 9 is depicting the fitness function which is used by meta-heuristic techniques to check that the solution is optimized or not at various levels in the form of makespan along with computation cost.

$$F_X = \frac{1}{Datacenter * VMj}\left[\sum_{i=1}^{DCi}\sum_{j=1}^{VMj}\frac{1}{VM}\frac{CPU\ Utilized}{CPUij} + \frac{Memory\ Utilized}{Memoryij} + \frac{Makespan\ Utilized}{Makespanij} + \frac{Bandwidth\ Utilized}{Bandwidthij}\right] \quad (9)$$

Equations 7 and 8 help in the calculation of the total processing cost represented in equation 6.

V. Simulation Results and Discussion

Four scenarios and a set of 10, 20, and 30 virtual machines have been set in CloudSim for simulation. To evaluate the performance of the proposed algorithm HC-CSO, the simulation results were compared with standard CSO and H-CSO. The makespan results concerning all scenarios are displayed in Table III. The results shown in Table III are the average of the results retrieved by the proposed algorithm HC-CSO along with other algorithms which were executed several times.

Simulation results of the proposed algorithm HC-CSO, H-CSO, and standard CSO for the scientific dataset Cybeshake_1000 are displayed in Fig. 3. Southern California Centre collected data and made the CybeShake workflow to analyze seismic hazards. For execution, this workflow requires almost near to lower CPU power and memory. It can be seen that at the x-axis 10, 20, and 30 VMs are being displayed and makespan at the y-axis. The results express that the HC-CSO outperforms other algorithms better local search after adding the Crow Search Algorithm and a better balance between seeking and tracing mode.

It is seen in Fig. 4 that the experiment carried out with the Montage_1000 dataset needs less memory and CPU power as compared to other workflows. The Montage dataset is having astronomical images collected and stored by NASA. The graph tells that the group of 10, 20, and 30, VMs as well as makespan, are shown at the x-axis and y-axis correspondingly. The results depiction tells the proposed algorithm HC-CSO performed efficiently as compare to other algorithms due to better convergence in less number of iterations.

TABLE III. Makespan Evaluation (in Sec)

Scenarios	VMs	CSO	H-CSO	HC-CSO
Scenario - 1 CyberShake_1000	10	4603.23	4520.45	4413.65
	20	3693.52	3500.27	3425.29
	30	2401.29	2350.15	2220.15
Scenario – 2 Montage_1000	10	2310.70	2270.09	2210.43
	20	2021.13	1982.05	1813.58
	30	1308.28	1219.01	1187.67
Scenario – 3 Inspiral_1000	10	52300.41	50940.98	49859.35
	20	40350.03	39241.18	37174.27
	30	23975.57	22548.23	20140.03
Scenario – 4 Sipht_1000	10	32820.02	31107.81	30740.41
	20	25203.13	23293.24	22980.37
	30	19113.29	18279.02	18033.13

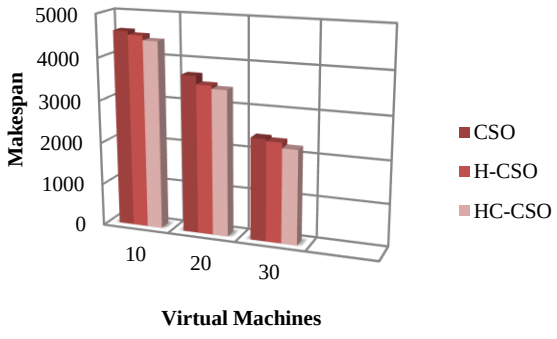

Fig. 3. Makespan Comparisons for CyberShake_1000 Tasks.

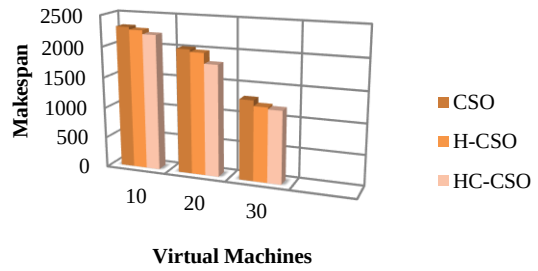

Fig. 4. Makespan Comparisons for Montage_1000 Tasks.

Fig. 5 is demonstrated the experimental results of the proposed algorithm named HC-CSO along with other algorithms like CSO and H-CSO for the dataset Inspiral_1000 which is related to the physics field and used to analyze the gravitational waves. This dataset needs high powered CPU and a large amount of memory for execution. The results concluding here that the proposed algorithm performed better than other algorithms because the HC-CSO algorithm has both capabilities of global and local searching as well as the algorithm also manage the velocities outrange. This capability of the proposed algorithm manages under-loaded and overload machines effectively by task migration. In this graph, the numbers of VMs are being displayed at the x-axis and makespan at the y-axis.

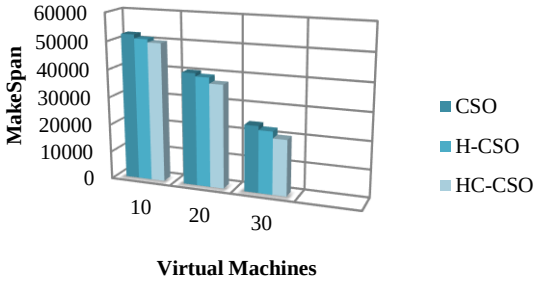

Fig. 5. Makespan Comparisons Inspiral_1000 Tasks.

Fig. 6 is displaying the results of the Shipt_1000 workflow processed with the proposed HC-CSO algorithm and others.

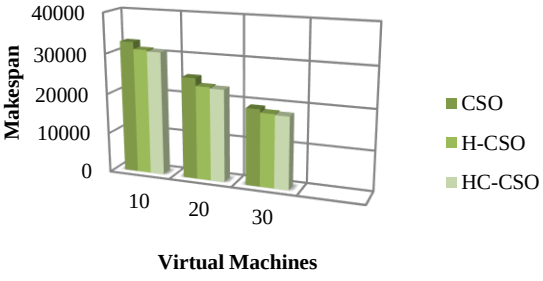

Fig. 6. Makespan Comparisons for Sipht_1000 Tasks.

For displaying the 10, 20, and 30 sets of VMs and makespan, the x-axis and y-axis are being used respectively. The Sipht dataset is used to represent sRNA-encoding genes of several bacteria and it has been released by the HIB (Harvard International Bioinformatics) Centre. This dataset requires huge memory and a high computational CPU. For the processing of this dataset, the proposed algorithm HC-CSO again worked well as compared to standard CSO and the H-CSO because the proposed algorithm chooses the most appropriate VM instead of high or low power due to better properties of local as well as global searching and the HC-CSO also avoids unnecessary diversity. The proposed algorithm could not trap in local minima due to Crow Search Algorithm local searching property.

In the last, it is concluded that the HC-CSO algorithm is giving better results in the form of a better makespan for all scenarios in comparison to standard CSO, and H-CSO. For all the scenarios, HC-CSO works better than CSO and, H-CSO because of the better pre-processing of tasks by HEFT, a good balance between seeking and tracing modes due to the Crow Search algorithm. The HC-CSO algorithm along with the Crow Search Algorithm searches the VMs at a local and global level very carefully to optimize the results in the minimum number of iterations. The SMIW method restricts the Cats to go outside the search space.

Table IV is summarizing the evaluation of the processing cost for the execution of all scenarios designed for simulation.

TABLE IV. COST CONSUMPTIONS (IN INDIAN RUPEES)

Scenarios	VMs	CSO	H-CSO	HC-CSO
Scenario - 1 CyberShake_1000	10	484.53	459.43	447.89
	20	601.43	590.29	567.78
	30	703.47	668.13	640.25
Scenario – 2 Montage_1000	10	181.83	167.53	159.13
	20	191.19	184.19	173.19
	30	198.13	190.89	169.17
Scenario – 3 Inspiral_1000	10	3663.48	3629.55	3413.42
	20	3759.37	3685.87	3543.65
	30	5430.17	5217.58	5130.71
Scenario – 4 Sipht_1000	10	2811.93	2759.19	2645.13
	20	2973.87	2918.13	2703.29
	30	3353.29	3109.24	2999.43

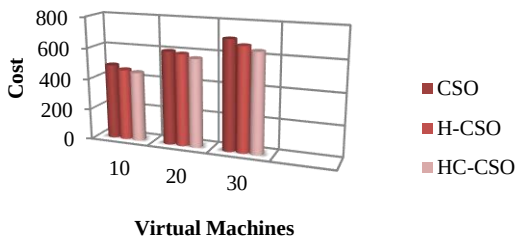

Fig. 7. Cost Comparisons for CyberShake_1000 Tasks.

For the first scenario having CybeShake_1000 dataset, a group of 10, 20, and 30 VMs along with processing cost are demonstrated at the x-axis and y-axis respectively in Fig. 7. The graph is expressing that the proposed HC-CSO algorithm is consuming minimal costs as compared to other algorithms. This is because the proposed algorithm has faster convergence and the positions of the workflows tasks on various VMs are updated smartly.

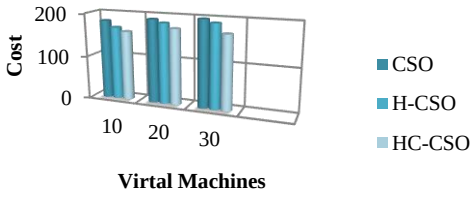

Fig. 8. Cost Comparison for Montage_1000 Tasks.

Fig. 8 is summarizing the processing cost consumption for the proposed algorithm HC-CSO, H-CSO, and standard CSO algorithms while working on 10, 20, and 30 VMs for the Montage_1000 dataset. The x-axis and y-axis of the graph are demonstrating the VMs and computing costs respectively. The proposed algorithm works better than H-CSO and CSO because the workflow tasks are migrated effectively for under-loaded virtual machines irrespective of their computing capacity.

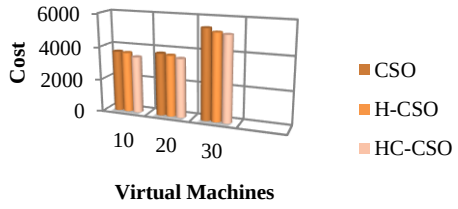

Fig. 9. Cost Comparison for Inspiral_1000 Tasks.

The processing cost results are being depicted in Fig. 9 for the Ispiral_1000 dataset. It can be seen VMs and Processing at the x-axis and y-axis respectively in the graph. The HC-CSO algorithm consumes less cost while executing the Ispiral_1000 dataset with all sets of VMs in comparison to other algorithms, this is because; the proposed algorithm makes an effective balance between seeking and tracing modes due to CSA integration. The VMs were picked up for execution of tasks irrespective of their MIPS, RAM, and bandwidth.

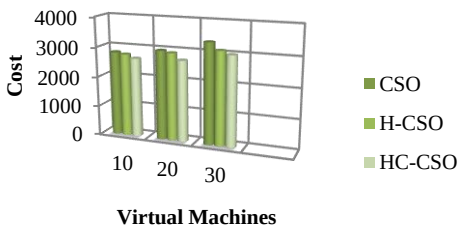

Fig. 10. Cost Comparisons for Sipht_1000 Tasks.

Fig. 10 is describing the results of processing cost obtained by the proposed HC-CSO, CSO, and H-CSO algorithms with various groups of virtual machines for the Sipht_1000 dataset. The VMs and processing cost can be seen at the x-axis and y-axis correspondingly in Fig. 10. Again, for the Sipht_1000 workflow; the proposed algorithm outperforms CSO and H-CSO in terms of computing cost. It is because the global and local searching properties of the proposed algorithm are balanced and the overloaded VMs loads are migrated to other VMs very smartly in the minimum time.

With these results, it can be specified that the proposed HC-CSO algorithm is better than other algorithms like CSO and H-CSO in respect of makespan and processing cost. The reason behind this is the good combination of global and local due to CSA. The Self-Motivated Inertia Weight factor integration overcomes the velocity outrange problem of Cats at tracing mode. The proposed algorithm chooses the virtual machines which are idle, under-loaded, or overloaded for workflow tasks migration among different VMs irrespective of their computing power, RAM, and Bandwidth.

VI. Conclusion

In the current technological era, cloud computing is one of the important emerging technologies used to store a large volume of data and other computing services in various science and technological fields. For computing facilities, various heuristic and meta-heuristic techniques have been developed. In this paper, an intelligent workflow scheduling algorithm named HC-CSO has been proposed to solve the workflow task scheduling problem. The proposed algorithm is a combination of H-CSO and the Crow Search algorithm. The H-CSO algorithm is an integration of HEFT and SMIW methods. The HEFT algorithm pre-processed the workflows tasks and initialized the proposed HC-CSO algorithm with these tasks. This process saves time and optimizes the results in a fewer number of iterations. The proposed HC-CSO algorithm didn't get trapped in local optima due to the good local searching capacity of the CSA. Velocity outranges cause to push the Cats outside the search space and affects the performance of the algorithm. The Self-Motivated Inertia Method overcame this problem.

The proposed HC-CSO algorithm outperformed CSO and H-CSO in terms of makespan and computing cost with all four scenarios having four scientific datasets CyberShake, Montage, Inspiral, Sipht, on a group of 10, 20, and 30 VMs. This is because the proposed algorithm chose the best VM among a group of VMs using its perfect global and local searching capacity for workflows task scheduling. The proposed approach is a generalized algorithm and will perform well with all types of scientific datasets despite a particular one.

In the future, the proposed algorithm can be tested at a wide scale to reduce makespan, cost and, other parameters in the cloud system and many other fields. The efficiency of the proposed algorithm can also be tested for independent tasks. A new technique can also be developed to enhance cloud performance.

REFERENCES

[1] G. Danlami, S. I. Abdul, Z. Anazida, Z. Zalmiyah, and A. Ahmad, "Hybrid Cat Swarm Optimization and Simulated Annealing for Dynamic Task Scheduling on Cloud Computing Environment," Journal of Information and Communication Technology, vol. 17, no. 3. pp. 435-467, July 2018.

[2] B. Saurabh, S. Santwana, and D. Madhabanda, "A Multi-Objective Cat Swarm Optimization Algorithm for Workflow Scheduling in Cloud Computing Environment," Internation Journal of Soft Computing, vol. 10, pp 17-45, 2015.

[3] B. Saurabh, S. Santwana, and D. Madhabanda, "Workflow Scheduling in Cloud Computing Environment using Cat Swarm Optimization," IEEE International Advance Computing Conference, pp. 680-685, 2014.

[4] G. Danlami, S. I. Abdul, Z. Anazida, Z. Zalmiyah, and A. Ahmad, "Cloud Scalable Multi-Objective Task Scheduling Algorithm for Cloud Computing using Cat Swarm Optimization and Simulated Annealing," IEEE International Conference on Information Technology, pp. 1007-1012, 2017.

[5] D. Negar, and J. N. Nima, "A Hybrid Particle Swarm Optimization and Hill Climbing Algorithm for Task Scheduling in the Cloud Environments," ICT Express, vol. 4, pp. 199-202, 2017.

[6] S. A. Thanna, A. S. Ashraf, and D. Yassine, "Binary PSOGSA for Load Balancing Task Scheduling in Cloud Environment," International Journal of Advanced Computer Science and Applications, vol. 9, no. 5, pp. 255-264, 2018.

[7] S. Narinder, and S. B. Singh, "Hybrid Algorithm of Particle Swarm Optimization and Grey Wolf Optimizer for Improving Convergence Performance," Journal of Applied Mathematics, Hindawi, vol. 2017, pp. 1-15, 2017.

[8] S. Heba, N. Heba, S. Walla, and M. H. Hany, "IPSO Task Scheduling Algorithm for Large Scale Data in Cloud Computing Environment," IEEE Access, vol. 7, pp. 5412-5420, 2019.

[9] P. Suraj, W. Linlin, M. G. Siddeswara, and B. Rajkumar, "A Particale Swarm Optimization-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," IEEE International Conference on Advanced Information Networking and Applications, pp. 400-407, 2010.

[10] E. Fatemeh, and M. B. Seyed, "A PSO Based Task Scheduling Algorithm Improved using a Load-Balancing Technique for the Cloud Computing Environment," Concurrency and Computation Practice and Experience, Wiley, Special Issue, pp. 1-16, Dec. 2017.

[11] Z. Zhou, C. Jian, H. Zhigang, Y. Junyang, and L. Fangmin, "A Modifuled PSO Algorithm for Task Scheduling Optimization in Cloud Computing," Concurrency and Computation Practice and Experience, Wiley, Special Issue, pp. 1-11, Sept. 2018.

[12] M. Neha, and S. Gaurav, "Modified Particale Swarm Optimization based upon Task Categorization in Cloud Environment," International Journal of Engineering and Advanced Technology, vol. 8, pp. 67-72, 2019.

[13] A. T. Medhat, E. Ashraf, E. K. Arabi, and A. T. Fawzy, "Cloud Task Scheduling Based on Ant Colony Optimization," International Conference on Computer Engineering and Systems, IEEE, pp. 64-69, 2013.

[14] S. Khurana, and R. K. Singh, "Workflow Scheduling and Reliability Improvement by Hybrid Intelligence Optimization Approach with Task Ranking," EAI Endorsed Transactions on Scalable Information Systems, Nov. 2019.

[15] M. S. Raja, P. Sanchita, and K. Abhishek, "Task Scheduling in Cloud Computing: Review," International Journal of Computer Science and Information Technologies, vol. 5, no. 6, pp. 7940-7944, 2014.

[16] L. Zhizhong, Q. Jingxuan, P. Weiping, and C. Hao, "Effective Task Scheduling in Cloud Computing Based on Improved Social Learning Optimization Algorithm," International Journal of Online and Biomedical Engineering, vol. 13, no. 6, pp. 4-21, 2017.

[17] A. S. M. Mohammed, G. Radhamani, A. G. H. Mohamed, and H. H. Syed, "Adaptive Cost-Based Task Scheduling in Cloud Environment," Scientific Programming, Hindawi, vol. 2016, pp. 1-9, 2016.

[18] A. Ali, and A. O. Fatma, "Task Scheduling Using PSO Algorithm in Cloud Computing Environments," International Journal of Grid Distribution Computing, vol. 18, no. 5, pp. 245-256, 2015.

[19] P. Guang, and W. Katinka, "Efficient Task Scheduling in Cloud Computing using an Improved Particle Swarm Optimization Algorithm," International Conference on Cloud Computing and Service, pp. 58-67, 2019.

[20] C. Hongyan, L. Xiaofei, Y. Tao, Z. Honggang, F. Yajun, and X. Zongguo, "Cloud Service Scheduling Algorithm Research and Optimization," Security and Communication Networks, vol. 2017, pp. 1-7, 2017.

[21] M. A. Aram, A. R. Tarik, and A. M. S. Soran, "Cat Swarm optimization Algorithm: A Survey and Performance Evaluation," Computational Intelligence and Neuroscience, vol. 2020, pp. 1-17, 2020.

[22] A. Alireja, "A Novel Metaheuristic Method for Solving Constrained Engineering Optimization Problems: Crow Search Algorithm," Computer and Structures, vol. 169, pp. 1-12, 2016.

[23] K. Mala, and S. Sarabjeet, "A Review of Metaheuristic Scheduling Techniques in Cloud Computing, Egyptian Informatics Journal, vol. 16, pp. 275-295, 2015.