# Independent Task Scheduling in Cloud Computing using Meta-Heuristic HC-CSO Algorithm

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

***Abstract*—Cloud computing is a vital paradigm of emerging technologies. It provides hardware, software, and development platforms to end-users as per their demand. Task scheduling is an exciting job in the cloud computing environment. Tasks can be divided into two categories dependent and independent. Independent tasks are not connected to any type of parent-child concept. Various meta-heuristic algorithms have come into force to schedule the independent tasks. In this, paper a hybrid HC-CSO algorithm has been simulated using independent tasks. This hybrid algorithm has been designed by using the HEFT algorithm, Self-Motivated Inertia Weight factor, and standard Cat Swarm Optimization algorithm. The Crow Search algorithm has been applied to overcome the problem of premature convergence and to avoid the H-CSO algorithm getting stuck in the local fragment. The simulation was carried out using 500-1300 random lengths independent tasks and it was found that the H-CSO algorithm has beaten PSO, ACO, and CSO algorithms whereas the hybrid algorithm HC-CSO is working fine despite Cat Swarm Optimization, Particle Swarm Optimization, and H-CSO algorithm in the name of processing cost and makespan. For all scenarios, the HC-CSO algorithm is found overall 4.15% and 7.18% efficient than the H-CSO and standard CSO respectively in comparison to the makespan and in case of computation cost minimization, 9.60% and 14.59% than the H-CSO and the CSO, respectively.**

***Keywords*—Crow search algorithm (CSA); cat swarm optimization (CSO); H-CSO algorithm; HC-CSO algorithm; heft algorithm; SMIW (self-motivated inertia weight); independent tasks; particle swarm optimization (PSO); QoS (Quality of Service); virtual machines (VMs)***

## I. Introduction

At present, the IT industries having cloud infrastructure are providing on-demand services to their customers [1][2]. These services may include hardware storage, memory, software, applications development at remote locations [3]. To fulfill these services, cloud service providers provide virtual machines to the users in order to execute their tasks. It is necessary to map all the tasks on each virtual machine carefully to optimize the performance of the cloud [4][5]. A service level agreement needs to be signed by the user and the service provider. According to that agreement various QoS (Quality of Service) parameters need to be decided before starting a service. These QoS may be budget, deadline of all tasks, security, throughput, etc. Quick execution of all tasks is highly demanded [5]. Task scheduling always remains a burning topic of research in cloud computing due to its NP-Hard properties [6][7]. The most critical problem of a cloud is to schedule the tasks to the perfect resources [8][9]. The cloud service providers apply many techniques to reduce the makespan and cost of tasks scheduling. There may be two kinds of scheduling techniques: independent scheduling and dependent scheduling. In the case of dependent scheduling, the tasks are interconnected with each other in the form of a workflow. In independent scheduling the tasks are not dependent on each other; they are autonomous as per their nature [6]. Further, the scheduling policy can be classified into two categories static scheduling and dynamic scheduling. In static scheduling, the amount of the data is known before execution but in the event of dynamic scheduling, the amount of the data is not known [10]. Scheduling is an important issue to increase the performance of a cloud. Many scheduling meta-heuristic methods have entered the Information Technology market. For example, Ant Colony Optimization, Genetic Algorithm, Particle Swarm Optimization, Gravitation Search Algorithm, etc. [11]. In this research, various algorithms have been utilized and these are described as:

*1) HEFT* – The HEFT algorithm [10] is having two phases: setting a task priority and selecting a virtual machine. In the priority phase, the HEFT sets the ranks of all the tasks. In the virtual machine selection phase, the HEFT calculates the earliest finish time of all tasks on the VMs. After this in HEFT, the tasks are set in decreasing order of their ranks and assign to virtual machines.

*2) Crow Search Algorithm* – The CSA algorithm [8] is another meta-heuristic technique. This algorithm has been developed by considering the real behavior of the Crow bird. The Crow is one of the most intelligent birds in this universe. Its memory is very sharp and it can easily remember the thing for a long time. One of the characteristics of the Crow is to steal the food of other birds by chasing them cleverly. Based on the above characteristic, the algorithm has been designed by A. Askarzadeh in 2016. According to the author [12], there are two states of the CSA algorithm, and both states are combined in equation 1.

$$X^{i,itr+1} = \begin{cases} X^{i,itr+1} + r_i \times fl^{i,itr} \times (m^{j,itr} - X^{i,itr}) & r_j \geq AP^{j,itr} \\ a\ random\ position & else \end{cases} \quad (1)$$

Where $r_i$ and $r_j$ are random variables between [0 – 1], fl denotes the flight length of the $Crow_i$, and $AP^{j,itr}$ is the awareness probability of the $Crow_j$ at the itr iteration.

In the first state, the local searching is taking place whereas in the second state the global searching (random searching). If the awareness probability is less than or equal to the $r_j$ variable then the global searching will be done otherwise local searching will be executed. The searching also depends on flight length (fl), a small value of fl will lead to local search. In above equation 1, it is also assumed that the $Crow_i$ will follow the victim $Crow_j$ and steal its food.

*3) Cat Swarm Optimization* – The CSO algorithm was designed by Chu and Tasi in the year 2006 based on two properties of a real cat, i.e. resting and hunting modes [13]. In the seeking mode, the cat moves freely in a random manner and remains alert. In the case of tracing mode, the cat has full of energy and moves towards the prey with high eagerness. Both the modes are described in detail in [14]. The pseudo-code [26] of the Cat Swarm Optimization is given in Fig. 1.

**Cat Swarm Optimization Algorithm**

```
1.  Randomly initialize Cats of Size N
2.  While Termination Condition Do
3.  Distribute Cats into Tracing Mode and Seeking Mode as per
    MR Flag
4.    For K = 1 to N
5.      Evaluation Cat Fitness
6.      If Cat_K is in Seeking Mode Then
7.        Execute Seeking Mode
8.      Else
9.        Execute Tracing Mode
10.     End If
11.   End For
12. End While
```

Fig. 1. Pseudo-Code of Standard Cat Swarm Optimization.

*4) H-CSO Algorithm* – This algorithm was formulated by using the HEFT algorithm original Cat Swarm Optimization, and the SMIW method. The SMIW method is given in the methodology adopted section in detail.

The flaws of the H-CSO algorithm are observed and the work has been done to overcome those limitations by inserting a local searching part of the standard Crow Search Algorithm [25]. Basically, the H-CSO algorithm is a hybrid of standard CSO, HEFT algorithm, and SMIW method. Some drawbacks of the H-CSO are described as:

- The H-CSO algorithm has poor local searching capacity. Although, the outrange drawback of the velocity formula of the tracing mode of the standard CSO has been removed in the H-CSO using the SMIW method.
- Due to the resting of Cats for a maximum time in the seeking mode of the H-CSO, it gets frozen in the local fragment.

The objective of this research is to simulate the HC-CSO [25] algorithm for independent tasks and make comparisons with H-CSO and other existing standard algorithms. The remaining information is specified in the methodology adopted section.

The remaining of the paper is organized as: In Section II related work has been presented. Section III is describing the methodology adopted and the simulation setup is explained in Section IV. Simulation results are discussed in Section V. Final Section VI is representing the conclusion and future scope of this research.

## II. Related Work

In this section, a study of various scientists is described. A hybrid CSO algorithm using the Simulated Annealing and Orthogonal Taguchi was designed by the scientists in [3]. The result analysis highlighted that the proposed method performs efficiently than MGA, MOACO, and MPSO for various QoS parameters. A task mapping approach using the PSO and Eagle Strategy was designed in [4]. The simulation results indicate that the proposed procedure is improved than the original PSO and other existing methods like RALBA, NMT-FOLS, etc. A joint task scheduling and resource placement policy was designed in the paper [5]. The makespan, degree of imbalance, resource utilization, and cost are improved using the newly designed policy as compared to existing GSO and GA methods. A new method named MHO was designed for task scheduling and load balancing in [6]. The proposed method having two phases MHOS-S and MHO-D were found better after results analysis as compared to other meta-heuristic methods. A new model for task allocation to virtual machines has been proposed in [8]. The experimental results summarized that the proposed ICSA algorithm reduced the makespan, waiting time, response time, and flow time as compared to FCFS and PSO methods. A Binary PSOGSA method was developed for the load balancing and task scheduling in the cloud in paper [11]. It is a bio-inspired load balancing algorithm used to manage the virtual machines for the load balancing issue. The outcome analysis demonstrated that the proposed method is efficient than originally developed Bin-LB-PSO and other techniques. An Average-Inertia Weight CSO algorithm (AICSO) was proposed in [14]. The simulation demonstrated that the AICSO is having a good convergence rate as compared to the standard CSO and ICSO algorithms. The authors proposed a cloud scheduling strategy named Genetic Algorithm-Chaos Ant Colony Optimization [15]. The proposed algorithm is optimal as compared to the ACO. A task scheduling strategy was designed using the PSO algorithm in the research paper [16]. The proposed method is a combination of Dynamic PSO and Cuckoo Search. It was identified by the results that the proposed algorithm worked fine than the original PSO. An adaptive cost-based method was developed in [17]. The proposed method was proved better by the simulation results in terms of various resources utilization like memory, bandwidth, CPU utilization, etc. A PSO-oriented load balancing method was proposed in [19]. The proposed method was designed by using a load balancing technique and was analyzed with traditional Round-Robin, the present PSO and load management method. The results were discussed by the authors and the new introduced technique was found efficient for balancing the cloud load. The paper [20] described that an improved PSO was developed using a discrete position updating strategy and the Gaussian Mutation operation. The proposed algorithm outperformed the existing algorithms. The researchers modified the PSO in [21] to overcome the problem

of slow convergence. The proposed PSO having dynamic inertia weight worked fine for cost reduction as compared to the existing IPSO, PSO-ACO, and standard ACO algorithms. An IPSO algorithm was proposed for task mapping on virtual machines in [22]. The process was achieved by splitting the coming tasks into many batches. The author in experiments summarized that the Improve PSO was efficient than existing Round-Robin, Honey Bee, and Ant Colony algorithms. A multi-objective CSO technique was framed in paper [23]. It was observed by the analysis of the results that the proposed MOCSO is very effective than MOPSO for better makespan, cost, etc. A CSO algorithm based on heuristic scheduling was introduced in the paper [24]. The results of this paper depicted that the proposed Cat Swarm Optimization worked better than the existing PSO.

From the above study, the main key points are found as follows:

*1)* The ACO algorithm is having a poor global search characteristic, due to that the convergence of the ACO is poor. The standard CSO jumps out from the search space due to the unbalanced tracing mode's velocity formula.

*2)* The PSO algorithm is famous for global searching and superior to the ACO for scheduling purposes.

*3)* The CSO algorithm is superior to the PSO due to its better convergence than PSO, but it gets trapped in local optima due to the resting behaviour of numerous cats in seeking mode.

*4)* H-CSO algorithm improved than the CSO but it may get trapped in the local search or in tracing mode while increasing the number of iterations.

## III. Methodology Adopted

To overcome the research gaps of the H-CSO described in the literature review section, a new algorithm has been designed named HC-CSO [25]. In the H-CSO, a local search portion of the CSA technique has been fused with the CSO. This technique balances the seeking and tracing modes which improve the searching capacity of the HC-CSO algorithm. The pseudo-code of the HC-CSO algorithm is given in Fig. 2.

As said earlier the HC-CSO method is a hybridization of the H-CSO and the Crow Search Algorithm's local searching portion. It has also been told earlier that the H-CSO is a combination of HEFT (described in the introduction section) and Self-Motivated Inertia Weight. The SMIW method is shown in equation 2.

$$\gamma = \gamma_{max} \times \exp\left(-c \times \left(\frac{itr}{itr_{max}}\right)^{c}\right) \quad (2)$$

Where, $\gamma$ is a weight factor inserted in tracing mode of the CSO whereas$\gamma_{max}$, and $c$ are constant factors greater than 1 i.e. 2.0.

As can be seen in Fig. 2 the local search portion of the CSA technique has been joined into the HC-CSO procedure. The functioning of the HC-CSO method is well-defined step by step as follows:

*1)* All the parameters are initialized at the beginning of the algorithm.

Some parameters used in the pseudo-code of the HC-CSO algorithm are: fl, $r_K$, $V_K$, c, $\gamma_{max}$, c1, MR flag, and number of iterations. The flight length of the crow is represented by the symbol fl. The Local and Global searching of the Crow Search Algorithm may be decided by the flight length. Flight length (fl) less than 1 is used for local searching, so, it has been set to 0.5. $r_K$ and r1 are random numbers between [0, 1]. $V_K$ is the initialized velocities of the cats. c and $\gamma_{max}$ are constant values that are set to 2. MR flag (Mixing Ratio) is set randomly [0.2-0.3] which means 20-30% of the cats are distributed in the tracing mode and rests are distributed in the seeking mode.

After this, the HEFT will calculate the rank based on the average execution time of all independent tasks and will arrange them as per their ranks.

*2)* If the solution is optimized at the beginning then, the algorithm will be terminated immediately and returned the solution else the population produced by the HEFT algorithm will be given for local searching via the CSA algorithm (as shown in line number 17 of pseudo-code).

*3)* The population generated by the CSA local search method will be handed over to the H-CSO algorithm for local as well as global searching via seeking and tracing modes. The cats are distributed in the seeking and tracing modes as per the Mixing Ratio (MR) parameter as described above. If the present $Cat_K$ is caught in the seeking mode then, the seeking mode will be executed otherwise the tracing mode will be processed.

*4)* In seeking mode, the S number of copies of the cats will be generated. Here, all cats are evaluated by the fitness function and the best cat is picked up randomly among various copies. After this, the current $Cat_K$ will be replaced by the best cat.

*5)* In tracing mode, the velocity of the current $Cat_K$ is updated by the modified velocity equation using the SMIW method (given in line number 28 in pseudo-code). Then, the position of the current $Cat_K$ is updated. Now, the evaluation of the Cats is taken place and the best cat is found out.

*6)* The best cat is saved in the memory.

*7)* This practice is sustained until the criterion is not met. In the end, the optimum solution is returned from the memory in the shape of the optimum result (Best Cat).

**HC-CSO 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) // Mapping of tasks

**BEGIN PROCEDURE**

1. Initialize fl (flight length), $r_K$, (velocity factor) $V_K$, c, $\gamma_{max}$, Coefficient c1, r1, MR flag, and no. of iterations
   /* Calculate Rank of independent tasks using HEFT Algorithm */
2. Feed the tasks in HEFT
3. **For** Each Task in List 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. $rank_u(t_i) = WAvg_i + \text{Max } t_j \in succ(t_i) (CAvg_{ij} + rank_u(t_j))$
   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. $X_{K,D} = X_{K,D} + r_K \times fl_{K,D} \times (M_{L,D} - X_{K,D})$
    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
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. $\gamma = \gamma_{max} \times \exp\left(-c \times \left(\frac{itr}{itr_{max}}\right)^c\right)$
    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 2.0
28. $V_{K,D} = \gamma \times V_{K,D} + \left(c1 \times r1 \times \left(X_{BEST,D} - X_{K,D}\right)\right)$
    Where, D = 1, 2, 3, …, 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 ($X_{BEST,D}$) having 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 Meta-Heuristic HC-CSO Algorithm.

## IV. Simulation Setup

For simulations of the independent tasks a computing machine was opted to have the configurations as Processor – Intel ® Core ™ i3-5005U 2.0 GHz speed, RAM – 4 GB, Hard Disk Drive – 1 TB, and Machine OS – Windows 10.

### A. Parameters

For experimental tests, a PowerDatacenter was generated having the formation as number of Hosts - 1, RAM – 25 GB, Each VM MIPS – 1000, Storage – 1 TB, and Bandwidth – 5000 bps. The cloud nature is heterogeneous and all details are displayed in Table I.

### B. Cost Plan

The cost plan for independent task scheduling is summarized in Table II. These charges will be applicable to the customer for using the datacenter services [27].

TABLE I. Simulation Parameters

| **PowerDatacenter** | |
|---|---|
| **Parameters** | **Values** |
| Number of Hosts | 1 |
| System Architecture | x86 |
| VMM | Xen |
| OS | Linux |
| Number of Cloudlets<br>Cloudlets Length Type | 500-1300 Independent<br>Random Length (300-700) |
| Numbers of VMs | 3, 5 and 8 |
| CPU (PEs Number) | 1 |
| RAM per VM | 512-1024 MB |
| Bandwidth | 500-1000 bps |
| Processing Elements per VM | 500 – 1000 MIPS |
| Image Size | 10000 MB |
| Policy Type | Time Shared |
| **PSO** | |
| No. of Particles | 100 |
| Max. Iterations | 300 |
| Weights C1 and C2 | 1.5 |
| **Standard CSO and H-CSO** | |
| No. of Cats | 100 |
| Max. Iterations | 300 |
| Weights (C1) | 1.5 |
| r1and $r_k$ (Random Variables) | [0, 1] |
| Mixed Ratio Percentage | Random Range [0, 1] |
| **HC-CSO** | |
| No. of Cats | 100 |
| Max. Iterations | 300 |
| Weights (C1) | 1.5 |
| r1(Random Variable) | [0, 1] |
| Mixing Ratio (MR) Percentage | Random Range [0, 1] i.e. 0.2-0.3 |
| fl (Flight Length) | 0.5 |

TABLE II. Cost Plan (In Indian Rupees)

| **Resource** | **Processor** | **RAM** | **Storage** | **Bandwidth** |
|---|---|---|---|---|
| Size | 500-1000 MIPS | 512 MB | Unlimited | 1000 bps |
| Cost | Rs. 3.00 per processor | Rs. 0.05 per MB | Rs. 0.1 | Rs. 0.10 per MB |

### C. Dataset Used

In the simulation studies, a group of 500, 800, and 1300 independent tasks having random lengths were opted and submitted to the VMs for execution.

### D. Performance Metrics

The performance metrics taken for this research are described below:

*1) Makespan:* The makespan [18] is stated as the maximum finished time occupied for the accomplishment of the last task in a set. It is the utmost generally used and very important parameter to map the performance of any algorithm in task scheduling. The makespan is calculated by the formula specified in equation 3.

$$\text{Makespan} = \max (CT_i)_{\ ti \in tasks} \quad (3)$$

Where, $CT_i$ is the completion time of $task_i$

*2) Computation cost:* The computation cost should be as less as possible; this is not only the demand of customers but; the industries also want the same to stay in the competitive IT market. The computation cost can be computed by equation 4.

$$Total\ Cost = \frac{MF+CF}{2} \quad (4)$$

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

$$MF = \frac{1}{No.\ of\ Hosts/Datacenters}\left[\sum_{x=1}^{VMx}\left(\frac{Number\ of\ Migrations}{Used\ VM}\right)\right] \quad (5)$$

$$CF = \sum_{x=1}^{VMx}\left(\frac{Processing\ Cost \times Memory\ of\ Tasks}{VM \times Datacenter}\right) \quad (6)$$

Equations 5 and 6 are utilized for the scheming of the total computation cost which is represented in equation 4.

*3) Fitness function:* Equation 7 is displaying how to calculate a fitness function that has been used in this research paper. It is used to check the optimization at various levels as shown in Fig. 2.

$$F_X = \frac{1}{Datacenter \times VMj}\left[\sum_{i=1}^{DCi}\sum_{j=1}^{VMj}\frac{1}{VM}\frac{CPU\ Utilized}{CPUij} + \frac{Memory\ Utilized}{Memoryij} + \frac{Makespan\ Utilized}{Makespanij} + \frac{Bandwidth\ Utilized}{Bandwidthij}\right] \quad (7)$$

## V. Simulation Results and Discussion

Previously, the HC-CSO algorithm was tested with scientific workflows [25]. Now, a set of three scenarios with 500-1300 independent tasks and a flock of 3, 5, and 8 VMs have been set in the CloudSim tool for the results analysis. The HC-CSO is compared with the PSO, CSO, and H-CSO algorithms. The simulation results in terms of makespan are shown in Table III. All algorithms were executed many times and the average results are displayed in the form of the makespan for each algorithm.

TABLE III. MAKESPAN ESTIMATION (IN SEC)

| Scenarios | VMs | PSO | CSO | H-CSO | HC-CSO |
|---|---|---|---|---|---|
| **Scenario - 1 500 Tasks** | 3 | 240.13 | 231.42 | 224.74 | 213.02 |
| | 5 | 225.29 | 213.27 | 203.89 | 195.24 |
| | 8 | 181.09 | 174.18 | 165.27 | 151.23 |
| **Scenario – 2 800 Tasks** | 3 | 425.35 | 417.21 | 401.93 | 387.54 |
| | 5 | 310.17 | 301.07 | 287.35 | 281.75 |
| | 8 | 279.85 | 263.17 | 255.03 | 243.29 |
| **Scenario – 3 1300 Tasks** | 3 | 722.13 | 712.28 | 697.13 | 681.08 |
| | 5 | 580.43 | 553.89 | 530.87 | 513.13 |
| | 8 | 452.24 | 437.23 | 433.11 | 400.39 |

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

Fig. 3 is representing that for 500 independent tasks the HC-CSO algorithm is performing better than standard PSO, CSO, and H-CSO algorithms on all 3, 5, and 8 VMs. This is because the convergence of the HC-CSO method has been improved due to better local searching and a balanced velocity factor by inserting the SMIW method.

In Fig. 4, it can be seen that the makespan is improved in the case of the HC-CSO method. In the case of 800 independent tasks, the HC-CSO method performs better than the PSO, CSO, and H-CSO algorithms due to a better exploration and exploitation rate and it is possible due to the integration of the CSA algorithm in H-CSO.

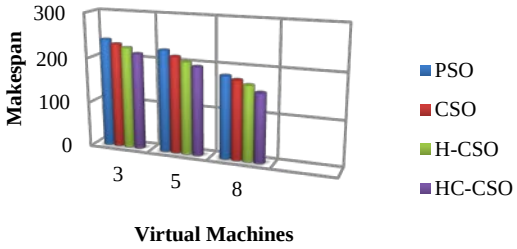


Fig. 3. Makespan Assessments for 500 Independent Tasks.

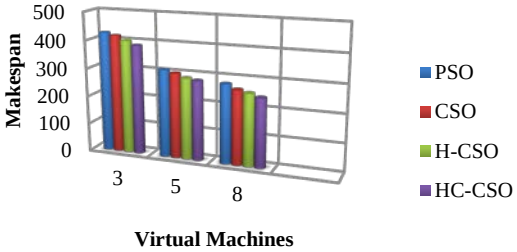


Fig. 4. Makespan Assessments for 800 Independent Tasks.

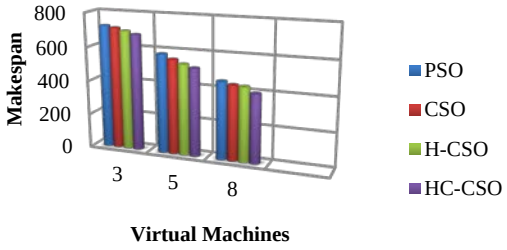


Fig. 5. Makespan Assessments for 1300 Independent Tasks.

Fig. 5 is justifying that the HC-CSO outperforms the PSO, H-CSO, and CSO with all flocks of VMs for 1300 independent tasks. The makespan is reduced due to better convergence of HC-CSO and a decent stability between seeking and tracing mode due to the collaboration of the CSA.

Table IV is briefing the cost consumptions for the execution of each algorithm with each scenario.

It can be seen that the virtual machines and cost are depicted at the x-axis and y-axis separately in Fig. 6, 7, and 8.

Fig. 6 is demonstrating that the cost is reduced by the HC-CSO algorithm with respect to all sets of VMs in the case of 500 tasks. Here, the HC-CSO is outperforming other algorithms depicted in Fig. 6 due to choosing of the best VM among all at a right time while mapping the independent tasks.

TABLE IV. COST CONSUMPTIONS (IN INDIAN RUPEES)

| Scenarios | VMs | PSO | CSO | H-CSO | HC-CSO |
|---|---|---|---|---|---|
| **Scenario - 1 500 Tasks** | 3 | 23.89 | 22.21 | 21.03 | 19.05 |
| | 5 | 30.23 | 28.20 | 27.52 | 28.29 |
| | 8 | 37.29 | 36.09 | 35.47 | 34.27 |
| **Scenario – 2 800 Tasks** | 3 | 41.13 | 38.33 | 37.08 | 33.51 |
| | 5 | 50.88 | 47.29 | 44.13 | 40.11 |
| | 8 | 65.58 | 62.13 | 59.15 | 54.01 |
| **Scenario – 3 1300 Tasks** | 3 | 70.03 | 67.51 | 64.53 | 59.24 |
| | 5 | 93.08 | 87.87 | 82.35 | 75.17 |
| | 8 | 117.13 | 113.25 | 103.89 | 85.87 |

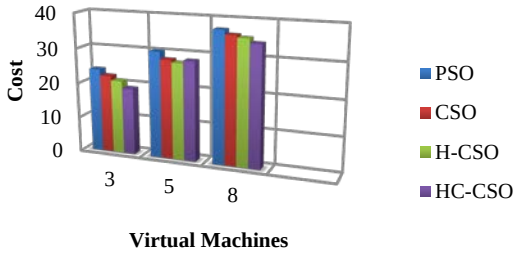


Fig. 6. Cost Assessments for 500 Independent Tasks.

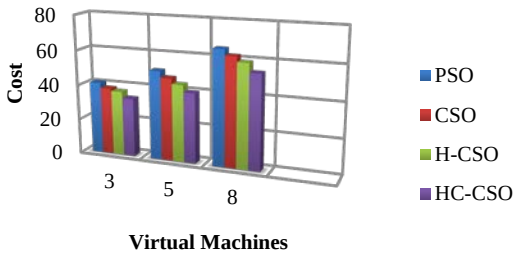


Fig. 7. Cost Assessments for 800 Independent Tasks.

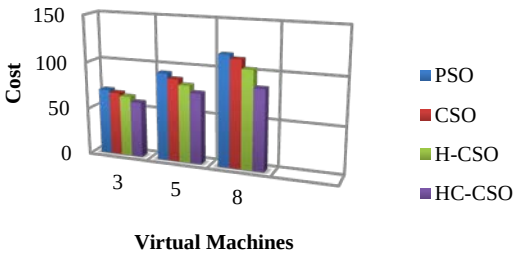


Fig. 8. Cost Assessments for 1300 Independent Tasks.

In Fig. 7, the performance comparison is done with 800 independent tasks on a flock of 3, 5, and 8 VMs. The HC-CSO algorithm won the race for reducing the cost as compared to the PSO, CSO, and H-CSO methods. It is due to the improvement of tracing mode by Self-Motivated Inertia Weight. The CSA gives the capacity of load adjustment to the HC-CSO algorithm in a better way by picking the best available virtual machine.

The HC-CSO algorithm is again working fine with 1300 tasks and a group of 3, 5, and 8 VMs. This can be observed in Fig. 8. The Cost is optimized here due to the good rate of task migration between under-loaded and over-loaded virtual machines. This happens because the HC-CSO algorithm is keeping a better global seeking property and local searching is improved by the CSA procedure. The outranged issue of the traditional Cat Swarm Optimization has been removed in the HC-CSO as well.

In the end, it is stated that the HC-CSO method has beaten all other techniques like PSO, H-CSO, and the traditional CSO concerning the makespan and computation cost due to better convergence, and the improved tracing mode. The Cats do not come out of the search space after inserting the SMIW method. The Self-Motivated Inertia Weight controls the velocity formula in tracing mode. So, it doesn't matter whether the search space is small or large. The CSA algorithm improves the exploration and exploitation of the H-CSO algorithm.

## VI. Conclusion and Future Scope

The cloud computing area is a burning topic of research nowadays. Many scientists have worked with GA, ACO, PSO, and CSO. It was seen that the CSO worked well in the name of makespan and processing cost. The scientists worked with dependent as well as independent tasks. The related work showed that the Cat Swarm Optimization worked better in many areas of cloud computing. In this research, a new algorithm named HC-CSO was utilized for experiments. This algorithm is a hybridization of three algorithms: HEFT, CSA, and H-CSO. The H-CSO was developed by the HEFT and SMIW formula.

After simulation, it was observed that the HC-CSO method outperformed the other techniques like the H-CSO, PSO, and CSO. The hybrid algorithm HC-CSO worked better with three scenarios having 500, 800, and 1300 independent tasks on 3, 5, and 8 virtual machines. After inclusion of all scenarios, it showed an overall 4.15% efficacy for makespan with minimization of 9.6% cost in comparison to the H-CSO and 9.60% efficacy with makespan for minimization of 14.59% cost than the CSO. The reason behind this good performance is the integration of the CSA algorithm in H-CSO. Also, the SMIW method controls the tracing mode velocity factor. This makes a check over the cats to escape out of the search space. It saves time as there is no need to initialize the velocity of the cats again and again.

This study proves that the HC-CSO technique is a generalized one as it has been tested on a different set of independent tasks. In the upcoming time, the HC-CSO algorithm can be tried for other objectives like energy consumption, resource utilization, load balancing, etc. Also, it can be applied in other areas of technology.

## References

[1] R. A. Al-Arasi, and A. Saif, "Task Scheduling in Cloud Computing Based on Meta-Heuristic Techniques: A Review Paper," EAI Endorsed Transactions on Cloud System, vol. 6, no. 17, pp. 1-19, 2020.

[2] A. M. Lonea, H. Tianfield, and D. E. Popescu, "Identity Management for Cloud Computing," New Concepts and Applications in Soft Computing, vol. 417, Springer, Berlin, pp. 175-199, 2013.

[3] D. Gabi, A. S. Ismail, A. Zainal, Z. Zakaria, and A. AI-Khasawneh, "Hybrid Cat Swarm Optimization and Simulated Annealing for Dynamic Task Scheduling on Cloud Computing Environment," Journal of ICT, vol. 17, no. 3, pp. 435-467, 2018.

[4] A. Kollu, and S. Vadlamudi, "Eagle Strategy with Cauchy Mutation Particle Swarm Optimization for Energy Management in Cloud Computing," International Journal of Intelligent Engineering & Systems, vol. 13, no. 6, pp. 42-51, 2020.

[5] D. Alboaneen, H. Tianfield, Y. Zhang and B. Pranggono, "A Meta-Heuristic Method for Joint Task Scheduling and Virtual Machine Placement in Cloud Data Centers," Future Generation Computer Systems, vol. 115, pp. 201-212, 2021.

[6] S. Peer Mohamed Ziyath, and S. Senthilkumar, "MHO: Meta Heuristic Optimization Applied Task Scheduling with Load Balancing Technique for Cloud Infrastructure Services," Journal of Ambient Intelligence and Humanized Computing, 2020.

[7] H. Singh, S. Tyagi and P. Kumar, "Scheduling in Cloud Computing Environment using Metaheuristic Techniques: A Survey," Emerging Technology in Modelling and Graphics, Advances in Intelligent Systems and Computing, vol. 937, pp. 753-763, 2020.

[8] A. Kousalya, P. Sinduja, V. H. Viswanathan, and Jeevitha, "Optimization of Task Scheduling using Improved Crow Search Algorithm in a Cloud Environment," International Journal of Pure and Applied Mathematics, vol. 119, no. 16, pp. 219-230, 2018.

[9] J. Natarajan, "Parallel Queue Scheduling in Dynamic Cloud Environment Using Backfilling Algorithm," International Journal of Intelligent Engineering & Systems, vol. 11, no. 2, pp. 39-48, 2018.

[10] A. Mazrekaj, A. Sheholli, D. Minarolli, and B. Freisleben, "The Experiential Heterogeneous Earliest Finish Time Algorithm for Task Scheduling in Clouds," 9th International Conference on Cloud Computing and Services Science, pp. 371-379, 2019.

[11] T. S. Alnusairi, "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.

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

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

[14] M. Orouskhani, M. Mansouri, and M. Teshnehlab, "Average-Inertia Weight Cat Swarm Optimization," Advances in Swarm Intelligence, Lecture Notes in Computer Science, vol. 6728, 2011.

[15] H. Cui, X. Liu, T. Yu, H. Zhang, Y. Fang and Z. Xia, "Cloud Service Scheduling Algorithm Research and Optimization," Security and Communication Networks, vol. 2017, pp. 1-7, 2017.

[16] A. Al-maamari, and F. A. Omara, "Task Scheduling Using PSO Algorithm in Cloud Computing Environments," International Journal of Grid Distribution Computing, vol. 8, no. 15, pp. 245-255, 2015.

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

[18] M. Kalra, and S. Singh, "A Review of Metaheuristic Scheduling Techniques in Cloud Computing," Egyptian Informatics Journal, vol. 16, pp. 275-295, 2016.

[19] F. Ebadifard, and S. M. Babamir, "A PSO-Based Task Scheduling Algorithm Improved Using a Load Balancing Technique for the Cloud Computing Environment," Concurrency and Computation Practice and Experience, Special Issue, pp. 1-16, 2017.

[20] G. Peng, and K. Wolter, "Efficient Task Scheduling in Cloud Computing using an Improved Particle Swarm Optimization Algorithm," 9th International Conference on Cloud Computing and Services Science, pp. 58-67, 2019.

[21] Z. Zhou, J. Chang, Z. Hu, J. Yu, and F. Li, "A Modified PSO Algorithms for Task Scheduling Optimization in Cloud Computing," Concurrency and Computation Practice and Experience, Special Issue, pp. 1-11, 2018.

[22] 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, pp. 5412-5420, 2019.

[23] S. Bilgaiyan, S. Sagnika, and M. Das, "A Multi-Objective Cat Swarm Optimization Algorithm for Workflow Scheduling in Cloud Computing Environment," International Journal of Soft Computing, vol. 10, no. 1, pp. 37-45, 2015.

[24] S. Bilgaiyan, S. Sagnika, and M. Das, "Workflow Scheduling in Cloud Computing Using Cat Swarm Optimization," IEEE International Advance Computing Conference, pp. 680-685, 2014.

[25] J. Bhagwan, and S. Kumar, "An HC-CSO Algorithm for Workflow Scheduling in Heterogeneous Cloud System," International Journal of Advanced Computer Science and Applications, vol. 12, no. 6, pp. 484-492, 2021.

[26] S. Goyal, S. Bhushan, Y. Kumar, A. H. S. Rana, M. R. Bhutta, M. F. Ijaz, and Y. Son, "An Optimized Framework for Energy-Resource Allocation in a Cloud Environment Based on the Whale Optimization Algorithm," Sensors, pp. vol. 21, no. 5, 1-20, 2021.

[27] 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, pp. 33-55, 2018.