IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

Scientific and Big Data Scheduling Using HC-CSO Algorithm in Large Scale Cloud IaaS

Jai Bhagwan[1], Sanjeev Kumar[2]

Assistant Professor, Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, India[1, 2]

Abstract: Cloud computing is a prominent area of research nowadays, because the cloud service providers are providing high scaled infrastructure based services on pay-per-usages. The services include not only storage, hardware but software and development platform as well. So, the IT industries are interested to move their business software and services to cloud based servers. This causes extensive data storage on Datacenters and need high computing power. Due to the rapid increase of users, the cloud execution time and cost can be compromised. So, scientists are developing scheduling and load balancing algorithms continuously for cloud datacenters. Various scheduling approaches have been designed so far such as ACO, PSO, GWO and CSO based swarm intelligence algorithms. In previous, research Cat Swarm Optimization algorithm performed better for task scheduling on cloud systems. However it has the problem of local optima stagnation. Due to that, various versions of CSO have been designed for scheduling. One of our designed HC-CSO algorithm performances are tested using large scale synthetic data for heterogeneous cloud environment in this research paper. The HC-CSO algorithm is a combination of SMIW inertia weight for seeking and tracing mode balancing and Crow Search Algorithm is injected for better exploration. The experiments are conducted using one Datacenter and 100 VMs of heterogeneous types by scheduling big-sized synthetic tasks. After evaluation, it has been found that the HC-CSO algorithm beats PSO, CSO, and GWO algorithms with respect to makespan, cost and degree of imbalance.

Keywords: Cloud Computing, Cat Swarm Optimization, HC-CSO Algorithm, Swarm Intelligence, Task Scheduling, Virtual Machines (VMs)

I. INTRODUCTION

The continuous growth of digital information has increased the demand of high computing infrastructure which is capable of processing large volume data efficiently. Applications in scientific research, health care, financial services, online transactions and others generate computational workloads that require high-performance computing resources. The traditional computing infrastructure faces various limitations in account of processing capacity, storage and resources scalability when dealing with large-scaled amount of data. As cloud computing is growing as an effective solution by providing on demand computational services through virtualization. It allows end users to execute large-scale application without maintaining dedicated infrastructure. However, the performance of the cloud environment depends on how the computational tasks are assigned or distributed among available resources. The scheduling problem becomes challenging in large-scale heterogeneous cloud environments where virtual machines have different capacity, memory, bandwidth and processing or execution cost. Selecting a virtual machine based on its processing capacity not always produce good results in efficiency and cost. So, efficient workload distribution can play a major role to influence the overall application execution time, cost and degree of imbalance.

Several research studies have shown the importance of resources heterogeneity and performance aware scheduling. Iosup *et al.* [1] investigated the performance of cloud services for scientific applications and obtained important performance measures related to virtualization and resource sharing. Abrishami *et al.* [2] examined deadline-based workflow scheduling in Infrastructure-as-a-Service (IaaS) clouds. The scientists targeted the need of balanced execution cost and application execution deadlines. Similarly, Arabnejad and Barbosa [3] developed an algorithm based on optimistic-cost-table-based scheduling for heterogeneous computing systems. They demonstrated the importance of task priorities and processors selection. Another important consideration is the relationship among computation and data movement. Data-intensive scientific workflows consist of multiple interdependent jobs or tasks that may require significant amounts of input and intermediate data. Ghafarian [4] studied cloud-aware scheduling using data-intensive workflows using volunteer computing resources and infrastructure. Malawski *et al.* [5] indicated that effective scheduling requires coordination between task execution and resource provisioning. The growth in complex scheduling environment has encourages the development and use of optimization methods. Particle swarm optimization, genetic algorithms, ant colony optimization and other swarm-based methods have been designed for

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2021.101133

effective task-to-resource assignments. Such techniques are particularly relevant due to the scheduling issues involve a large solution space and multiple objectives. Li *et al.* [6] developed a chaotic PSO-based algorithm for market-oriented workflows scheduling in cloud system. Mansouri *et al.* [7] designed a hybrid scheduling strategy based on modified PSO and fuzzy theory. These types of approaches demonstrate the potential of swam intelligence based optimization for task scheduling decisions in cloud systems. Despite considerable progress, a large-scale scheduling remains challenging because minimizing one parameter may negatively affect another. Assigning tasks to high capacity VMs may reduce the execution time but increase resources utilization cost. Selecting lower-cost resources may increase the makespan and response time. A balanced scheduling is also required because an unbalanced scheduling can leave some VMs under-utilized and others overloaded. In previous research [21], we have considered makespan as a major parameter for minimization and resources were selected based on their characteristics to reduce the cost as well using HC-CSO algorithm. That means, the tasks were distributed on virtual machines in such a way that utilizes the computing capacity (CPU in terms of MIPS), RAM and Bandwidth in an efficient manner to optimize the makespan and cost. But in this study, we are directly minimizing makespan, cost and degree of imbalance using a separate fitness function (discussed later). The main contributions of this paper are described below:

- Big data i.e. scientific workflows and synthetic tasks are considered to schedule on various heterogeneous virtual machines based on makespan, cost and degree of imbalance.
- For data scheduling HC-CSO [21] algorithm has been used which is a combination of crow search and cat swarm optimization. Exponential decay inertia weight named SMIW [22] has also been incorporated to balance the seeking and tracing modes.
- The HC-CSO algorithm out performed CSO, PSO and GWO with respect to makespan, cost and degree of imbalance.

The rest of the paper is structured as: section 2 describes the literature review, in section 3, system model is addressed, section 4 presents data and methods, section 5 explains the results and discussion and finally conclusion and future scope are explained in section 6.

II. LITERATURE REVIEW

Research on cloud scheduling has progressively evolved from conventional resource allocation to cost-effective and multi-objective optimization. Iosup *et al.* [1] evaluated cloud computing services for various tasks scientific applications and highlighted the influence of resource sharing and performance efficiency on scientific types of workloads. Abrishami *et al.* [2] proposed deadline-constrained workflow scheduling algorithms for IaaS clouds. The scientists have addressed the challenge of satisfying application deadlines while minimizing execution costs. Their work proved that cloud resource provisioning and task scheduling may be considered together in order to achieve economically effective schedules. Arabnejad and Barbosa [3] proposed the Predict Earliest Finish Time algorithm for heterogeneous computing systems. Their algorithm uses an optimistic cost table to improve task prioritization and processor or resource selection. Ghafarian [4] proposed a scheduling algorithm for workflows using volunteer computing resources and cloud infrastructure. The proposed method considered deadline and tried rescheduling tasks when resource could not meet their deadlines. Malawski *et al.* [5] examined the scheduling and resource provisioning for large workflows under budget and deadline constraints. This study considered both static and dynamic scheduling methods including uncertainty in task execution estimates. Metaheuristic optimization has also achieved considerable attention in cloud scheduling research. Li *et al.* [6] investigated a chaotic particle swarm optimization-based market-oriented task-level mapping in cloud system. The proposed approach combined chaotic optimization to improve the searching process for addressing computing cost. Mansouri *et al.* [7] developed a hybrid task scheduling method using fuzzy theory and modified PSO algorithm. They have incorporated different velocity update methods and population diversity mechanisms to improve scheduling search process. The research shows the usefulness of combining optimization strategies to improve task allocation to proper available resources. Data management and resources distribution have also been considered as essential scheduling factors. Bryk *et al.* [8] studied storage aware scheduling methods for tasks ensembles in cloud infrastructure. The study considered storage requirements and data movement while considering suitable resources. Praveen *et al.* [9] designed a resource allocation method using social group optimization for improvement of resource distribution and load balancing in heterogeneous cloud infrastructure. These studies specify that the scheduling process not only depends on processing capacity but also on storage availability and workloads distribution. Multi-objective scheduling has emerged as an important parameter for cloud scheduling research. Chen *et al.* [10] proposed a multiple-population based ant colony method for multi-objective cloud task scheduling. Their study considered simultaneous execution time of workflows and monetary cost along with maintaining search diversity.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Vol. 10, Issue 11, November 2021
DOI: 10.17148/IJARCCE.2021.101133

Hu *et al.* [11] designed an approach based on PSO algorithm for many objectives such as makespan, execution cost, reliability etc. Zhou *et al.* [12] investigated cost and makespan-aware scheduling strategies for hybrid cloud computing environment. This study considered allocation of workflows among private and public cloud resources. Kalra and Singh [13] designed a multi-criteria workflow scheduling under deadline and cost constraints. B. Kumar *et al.* [14] proposed a discrete binary CSO algorithm for workflows scheduling applications in cloud systems. In this research scientific workflows of different sizes are evaluated using DBCSO algorithm. The DBCSO algorithm outperformed BPSO and PSO algorithm with respect to makespan. S. Bilgaiyan *et al.* [15] presented CSO based scheduling algorithm for workflows in cloud systems. The authors scheduled a hypothetical workflow and compared the results of proposed CSO with PSO. CSO algorithm outperformed in terms of cost and load distribution. In 2020, research continued to report the growing complexity of cloud workflow scheduling. Hu *et al.* [16] proposed an intelligent scheduling method for big-data application. They considered response time and resource utilization as important parameters. Shirvani [17] developed a hybrid PSO algorithm for scientific workflow scheduling in heterogeneous distributed computing system. Pasdar *et al.* [18] did study for scientific workflows scheduling for data movement between private and public cloud infrastructure. Vakilinia [19] studied energy efficient and load aware resource allocation strategy for cloud datacenters and gave importance to workload variation and energy consumption. Cui *et al.* [20] performed a study using genetic algorithm and ant colony optimization. They considered makespan and flow-time as their scheduling objectives.

In previous research [21], we have designed a new hybrid scheduling algorithm called HC-CSO. In this algorithm, the SMIW [22] inertia weight was incorporated in tracing mode for balancing the seeking and tracing modes. Also, the exploration of the algorithm was improved using crow search algorithm based strategy [23]. The detail of the algorithm is also discussed in coming sections. This algorithm found helpful to remove the limitations of standard CSO i.e. stagnation in local optima and poor exploration [24].

III. SYSTEM MODEL

In this research, we have created 100 VMs in a host under one datacenter for simulation in CloudSim 3.0 using NeatBeans 8.0. We have considered 5 types of heterogeneous VMs having 500-1000 MIPS capacity, 512-1024 MB RAM and 1000 bps Bandwidth. The experiments were conducted on a computing machine having Intel ® Core™ i3-5005U 2.0 GHz processor, 4 GB RAM, 1 TB HDD and Windows 10.

3.1 Problem Definition

Let $T = \{T_1, T_2, T_3, T_4, T_5, \dots, T_n\}$ be the set of n tasks and $VM = \{VM_1, VM_2, VM_3, VM_4, VM_5, \dots, VM_m\}$ be the set of m virtual machines in the cloud environment. The scheduler of the cloud must ensure a mapping $f: F \rightarrow$ VM such that each task is assigned to suitable VMs for optimization. Figure 1 represents an example of 10 tasks and 3 VMs allocations.

Task-VM Mapping Representation					
VM1	T1	T3	T7	T9	T10
VM2	T4	T5	-	-	-
VM3	T2	T6	T8	-	-

Fig. 1 Task-VM Allocation Representation

3.2 Makespan Model

The makespan is the maximum finishing time consumed by cloud system to schedule a group of tasks [21] which is presented in Eq. (1).

$$Makespan = max(F_i) \tag{1}$$

Where, $i \in T$, T is set or group of all tasks and Fi is finish time of tasks i.

3.3 Cost Model

In real cloud computing system, a cloud service provider provides the infrastructure on pay-per-usage model. We have used a cost model [21][24] for considering the pay-per-usage concept. It is supposed a data-center has one host holding several VMs which may utilize the computation cost given in Eq. (2).

$$cost = \frac{CF+MF}{2} \tag{2}$$

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

The Migration factor (MF) is computed using Eq. (3). Let us say: HM is number of host machines, DC is number of data-centers, NMig is number of migrations of tasks and VM_{utl} is number of VMs utilized, then:

$$MF = \frac{DC}{HM} \times \frac{NMig}{VM_{utl}} \tag{3}$$

Computation cost factor is computed using Eq. (4). Let us say: V is the total number of VMs, PT_j is the processing time of VM_j, Mem_j is total task memory on VM_j, $MIPS_j$ is MIPS of VM_j and DC_{TMips} is total data-centers MIPS, then:

$$CF_{base} = \frac{1}{V} \sum_{j=1}^{V} \frac{PT_j \times Mem_j}{MIPS_j \times DC_{TMips}} \tag{4}$$

Here, we have used VM scaling factor to scale up the cost if the number of VMs is increased under pay-per-usage model, and it is calculated using Eq. (5). Let us say: w_{sc} is VM scaling weight which is decided by the cloud service provider in case of number of VMs is increased, then:

$$SF = 1 + w_{SC} \times VM_n \tag{5}$$

Final CF (Cost Factor) is computed using Eq. (6). Let us say: norm is normalization factor i.e. 100 to balance the cost for a realistic environment of cloud system.

$$CF = norm \times CF_{base} \times SF \tag{6}$$

3.4 DOI Model

Degree of imbalance (DOI) is a parameter by which load distribution can be estimated. This parameter is useful for load distribution calculated using Eq. (7).

$$DOI = (WL_{max} - WL_{min})/WL_{avg} \tag{7}$$

3.5 Fitness Function

The previous fitness function [21] is helpful in finding suitable VMs for workflows or synthetic tasks scheduling while minimizing the makespan and cost based on normalized makespan and VMs normalized characteristics. But in this research, we have considered a fitness function given in Eq. (8). This fitness function is directly involving makespan, cost and degree of imbalance (DOI) for scheduling objectives.

We have combined makespan, cost and DOI metrics in a single objective fitness function (F_x) which is computed using Eq. (8). The number of datacenters and VMs are used to normalize the fitness values. In this study, we are minimizing the values of makespan, cost and degree of imbalance for a fair workload distribution among suitable VMs.

$$F_x = (min) \frac{w_1 \times makespan + w_2 \times cost + w_3 \times DOI}{DC \times VMs} \tag{8}$$

Where, DC is number of datacenters, VMs are virtual machines, w_1, w_2 and w_3 are weights used to give the priorities to makespan, cost and DOI. Here, a slightly higher priority is given to makespan in order to ensure stable performance of the cloud system. The values of w_1, w_2 and w_3 are 0.5, 0.3 and 0.2 respectively given that $w_1 + w_2 + w_2 = 1$.

IV. DATA AND METHODS

This study considers the scheduling problem using real world Scientific Data CyberShake and Epigenomics containing 1000 tasks and 997 tasks respectively of various lengths and Synthetic type's jobs having 5000, 10000 and 20000 big length (5000-154900 MI) tasks.

4.1 HEFT Algorithm

HEFT algorithm [21][25], is a heterogeneous earliest finish time algorithm that is used to initialize the population or swam initially for all algorithms i.e. it is a path planning algorithm.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

Algorithm 1: HEFT Algorithm

Input: Workloads T (Workflows or Synthetic Independent Tasks), VMs V
Output: Initial Tasks Assignment to VMs

```
For each task t in T Do
    avg ← average(ET[t, vm] for all vm in V)
    If workflow Then
        rank(t) ← avg + max(rank(successors(t)))
    Else
        rank(t) ← avg
    End If
End For
Sort the tasks in descending order  //maintain critical task priory
For each solution S Do
    Set all VMs v finish times = 0
    For each task t in sorted list Do
        For each vm in V Do
            Est ← max(finish time of predecessors)
            ft ← Est + ET[t, vm]
        End For
        Choose vm with minimum finish time ft
        Assign task t to vm and update finish time ft
    End For
End For
return initialized population (initial tasks assignments to VMs)
```

This HEFT algorithm can work for workflows and independent tasks. In case of independent tasks, the task dependency will be nil. The ranking based on HEFT is assigned to the parent and child tasks available in workflows used in this research. The detailed steps are depicted in Algorithm 1 for HEFT policy.

4.2 Crow Search Based Strategy

Crow search algorithm is a population based searching algorithm used to solve various engineering optimization problems [23]. There are two phases of crow search algorithm a) random searching and b) local searching [21][23]. We have used local searching equation to exploration or refine the solutions for HC-CSO algorithm [21]. The local searching equation which was integrated in HC-CSO is given in Eq. (9).

$$X_i^{t+1} = X_i^t + r_i \times FL_i\left(M_j^t - X_i^t\right) \tag{9}$$

Where, X_i^t is the current position, FL_i is the flight length which is responsible for searching behaviour. A bigger flight length leads to global searching whereas the lower flight length leads to local searching, r_i and r_j are random variables between (0, 1), M_j is the memory or best hiding location of $crow_j$. The steps of the crow search based strategy are given in Algorithm 2.

Algorithm 2: Crow Search Based Strategy

Input: Current Solution Generated by HEFT Cat, Populations Pop, FL,
Output: Updated Task-VM Assignment (Cat position)

```
Randomly select a Crow from population Pop
For each dimension D in Cat position Do
    Generate a random number r ∈ [0, 1]
    Cat[D] ← Cat[D] + FL× r × (Crow_memory[D] - Cat[d]) // update position i.e. exploration
End For
return updated solution i.e. Cat position
```

4.3 HC-CSO Algorithm

The cat swarm algorithm is developed by Chu *et al.* [26] in 2006. This algorithm uses two modes seeking and tracing. Seeking mode is used for exploration and tracing mode is used for exploitation. The brief introduction of both phases is discussed here:

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

1) Seeking Mode: This mode is responsible for exploration using SMP (Seeking Memory Pool), SRD (Seeking Range Dimension), CDC (Count of Dimensions to Change), SPC (Self Position Consideration), and to distribute the cats in seeking and tracing modes MR (Mixing Ratio) is used. The steps involved in seeking mode are:
   - Make SMP copies of Q^{th} cat.
   - By keeping in mind CDC, perform the position update.

$$X_{Q,D} = (1 + rand \times SRD) \times X_{Q,D} \tag{10}$$

     Where, $X_{Q,D}$ is the position of the cat and rand is a random number between (0, 1).
   - Discover the best cat by fitness evaluation.
   - Replace the original cat with best copy.
2) Tracing Mode: In this mode, the cat updates the position faster for finding the food. The velocities and positions are updated using Eqs. (11) and (12) respectively.

$$V_{Q,D} = \omega \times V_{K,D} + \left(c1 \times r1 \times \left(X_{BEST,\,D} - X_{Q,D}\right)\right) \tag{11}$$

Where, $V_{Q,D}$ is velocity, c1 is leaning coefficient and r1 is a random number between (0, 1) in the basic CSO. Static inertia weight ω can be used for basic CSO algorithm which is normally set between 0.5 and 0.8.

$$X_{Q,D} = X_{Q,D} + V_{Q,D} \tag{12}$$

4.4 SMIW

Self-motivated inertia weight [21][22] or exponential decay inertia weight is responsible for local and global searching balance i.e. after using it in velocity equation, the seeking and tracing modes are balanced. This may results good performance. The SMIW is given in Eq. (13). The detail about this method is given in [22], in which it is suggested to use in PSO algorithm.

$$\omega = \omega_{max} \times \exp\left(-l \times \left(\frac{itr}{itr_{max}}\right)^{g}\right) \tag{13}$$

Where, ω_{max} is initial inertia value which is set to 2.0, l and g parameters are local and global searching attractors which are also set to 2.0. This adaptive inertia weight decreases with the time and control the premature convergence for a time. But for more effective convergence and exploration, crow search strategy plays a vital role. The effect can be understood by the results and discussion section. Algorithm 3 is describing the HC-CSO steps.

Algorithm 3: HC-CSO Algorithm

Input: Cats generated by HEFT i.e. solutions, SMP, CDC, MR, max_itr etc.
Output: Best Schedule gBest

```
For Gen = 0 to max_itr Do
    Update Seeking and Tracing Modes using MR value
    For each cat in Cats Do
        Call Algorithm 2 for Exploration  // position update for new solutions
        If cat is in Seeking Mode Then
            run Seeking Mode  // Update positions using Eq. (10)
        Else
            run Tracing Mode  // Update positions using Eq. (12), also use SMIW ω inertia weight
        End If
        If cat.fitness < gBest.fitness Then
            gBest ← copy (cat)
        End If
    End For
End For
return Best Schedule (gBest)
```

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

V. RESULTS AND DISCUSSION

This section discusses the results obtained by running all algorithms for several times (20 runs each) for a fair comparison. All the algorithms have been implemented for a transparent evaluation. The simulation parameters are given in Table 1. The performance parameters are makespan, cost and DOI which are already discussed in previous section.

TABLE 1 SIMULATION PARAMETERS

Algorithm	Parameter	Value
HC-CSO	Max Generations	200
	No. of Cats (Population)	50
	SMP, SRD, CDC, MR, FL	10, 0.3, 0.3, 0.2, 0.5 respectively
	c1, rand, ω_{max}, l, g	1.5, (0,1), 2.0, 2.0, 2.0 respectively
CSO	Max Iterations	200
	No. of Cats	50
	SMP, MR, SRD, CDC, c1	10, 0.2, 0.3, 0.3, 1.5 respectively
GWO	Max Iterations	200
	No. of Wolves	50
PSO	Max Iterations	200
	No. of Particles	50
	c1, c2, r1, r2	1.5, 1.5, (0, 1), (0, 1) respectively

5.1 First Scenario Results Discussion

In the first, scenario we have considered CyberShake and Epigenomics dataset for scheduling. The results obtained from these datasets are depicted in Figs. 2, 3 for CyberShake and Figs. 4 and 5 for Epigenomics.

For CyberShake dataset using 100 VMs, HC-CSO achieves the lowest average makespan of 767.94 Sec. outperforming PSO (3071.82 Sec.), GWO (5409.02 Sec) and CSO (1243.25 Sec.). It reduces makespan by 38.23% compared to CSO algorithm. HC-CSO algorithm is demonstrating more efficient tasks scheduling using 100 VMs. In case of cost, HC-CSO algorithm records the lowest average cost of 269.58 compared to CSO (418.99), PSO (1236.52) and GWO (4510.19). This represents a 35.66% cost reduction over CSO algorithm. The HC-CSO is more economical resource allocation technique found so far. HC-CSO obtains the lowest DOI of 3.04 compared to CSO (5.33), GWO (5.37) and PSO (13.38). The DOI reduction by HC-CSO algorithm indicates better load balancing across the available VMs. The convergence results in Fig. 3 show that HC-CSO achieves the lowest final average fitness compared to other algorithms. It converges rapidly during early iterations and maintains the best final fitness among all algorithms. This demonstrates the effective balancing of exploration and exploitation due to which it outperforms over other algorithms. For Epigenomics dataset using 100 VMs, HC-CSO achieves the lowest makespan of 238206.54 Sec., compared to CSO (320051.86 Sec.), PSO (688191.80 Sec.) and GWO (1024936.24 Sec.). HC-CSO algorithm reduces the makespan by 25.57% compared to CSO algorithm which is demonstrating better workflow execution-time optimization.

IJARCCE

ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

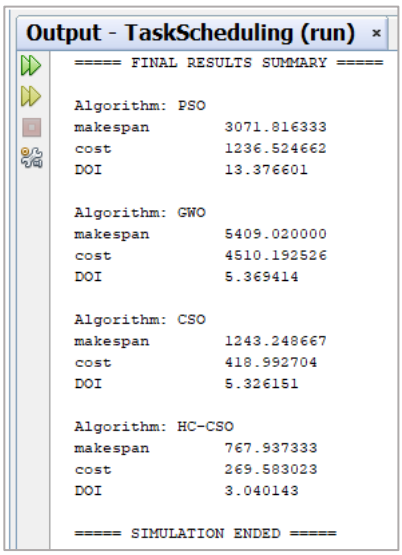


Fig. 2 Makespan, Cost and DOI Results for CyberShake

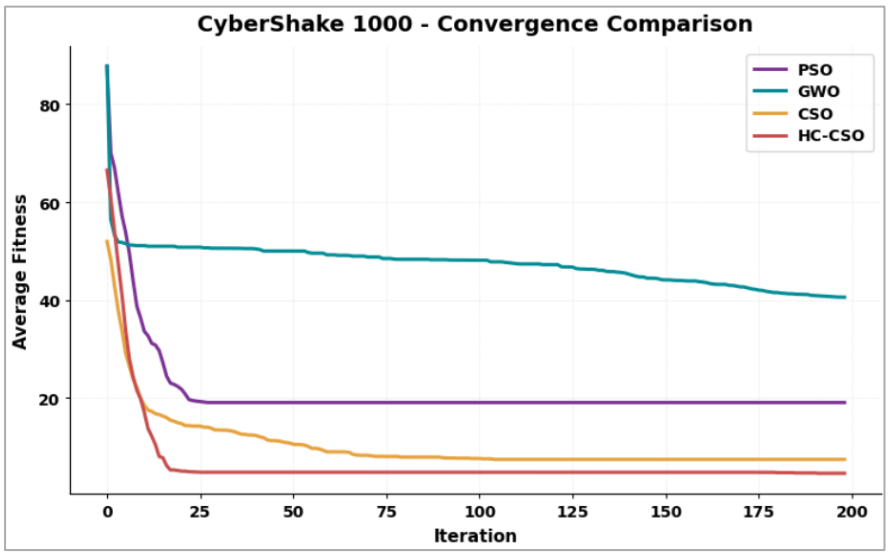


Fig. 3 Convergence Comparison for CyberShake

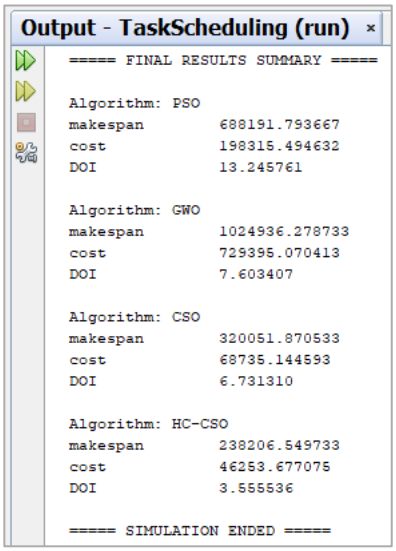


Fig. 4 Makespan, Cost and DOI Results for Epigenomics

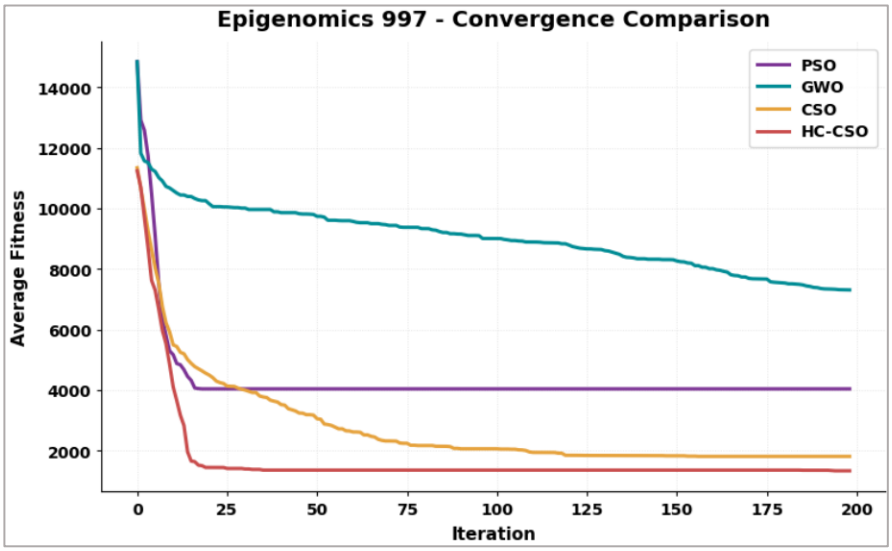


Fig. 5 Convergence Comparison for Epigenomics

For cost, HC-CSO records the lowest average cost of 46253.68 in comparison to CSO (68735.14), PSO (198315.50) and GWO (729395). This represents a 32.71% cost reduction over CSO algorithm. It is indicating more economical resource allocation. In case of DOI, HC-CSO gets the lowest average DOI of 3.56 in comparison to CSO (6.73), GWO (7.60 and PSO (13.25). It reduces the DOI by 47.18% compared to CSO algorithm. It is an indication of improved load balancing across available VMs. For convergence results shown in Fig. 5, it is obtained that HC-CSO algorithm gets the lowest final fitness among all algorithms. It converges rapidly during the initial iterations and maintains the lower fitness throughout the later stages or generations. This makes indication of effective optimization and strong balance between exploration exploitation compared to all other algorithms. Hence, it is proved that the Crow Search strategy helps for exploration i.e. finding new solutions effectively and SMIW inertia weight makes the balance between exploration and exploitation for a large scale scenario of 100 VMs.

5.2 Second Scenario Results Discussion

In this scenario, we have considered Synthetic Independent Tasks dataset of 10000 and 20000 big length tasks. The results are presented in Figs. 6 and 7 for 10000 tasks and in Figs. 8 and 9 for 20000 tasks. The tasks are having 5000 – 150000 MI length (random length).

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Vol. 10, Issue 11, November 2021

DOI: 10.17148/IJARCCE.2021.101133

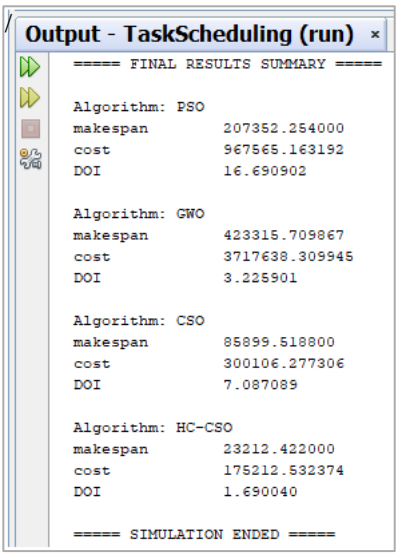


Fig. 6 Makespan, Cost, DOI Results for 10000 Tasks

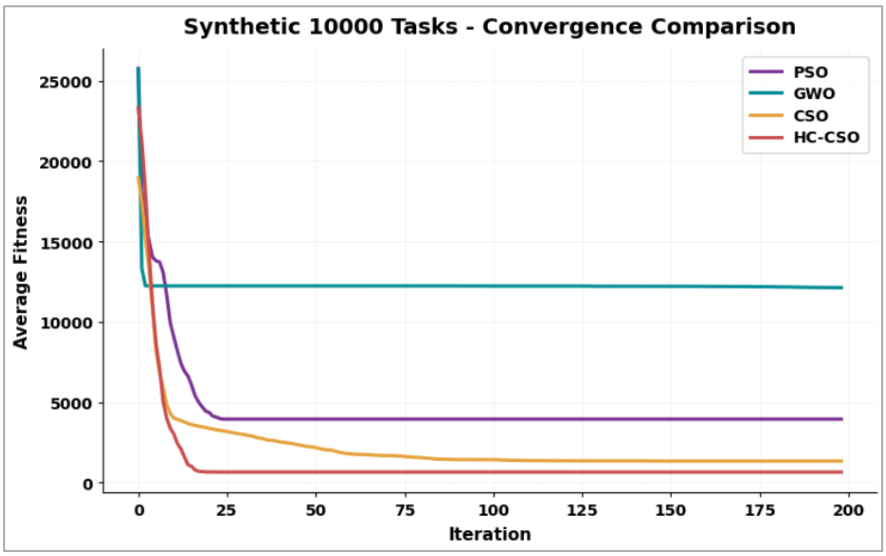


Fig. 7 Convergence Results for Synthetic 10000 Tasks

For Synthetic 10000 tasks, HC-CSO achieves the lowest makespan of 23212.42 Sec. followed by CSO (85899.52 Sec.), PSO (207352.25 Sec.) and GWO (423315.71 Sec.). HC-CSO algorithm reduces makespan by 72.98% compared to CSO. It is demonstrating that HC-CSO is effective in optimizing large length 10000 tasks for makespan. For cost, HC-CSO records the lowest execution or processing cost 175212.53 compared to CSO (300106.28), PSO (967565.16) and GWO (3717638.31).

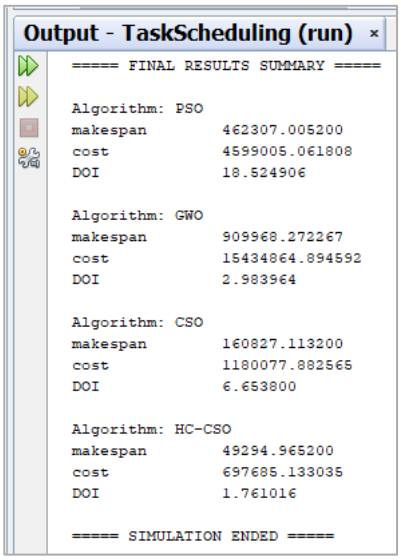


Fig. 8 Makespan, Cost, DOI Results for 20000 Tasks

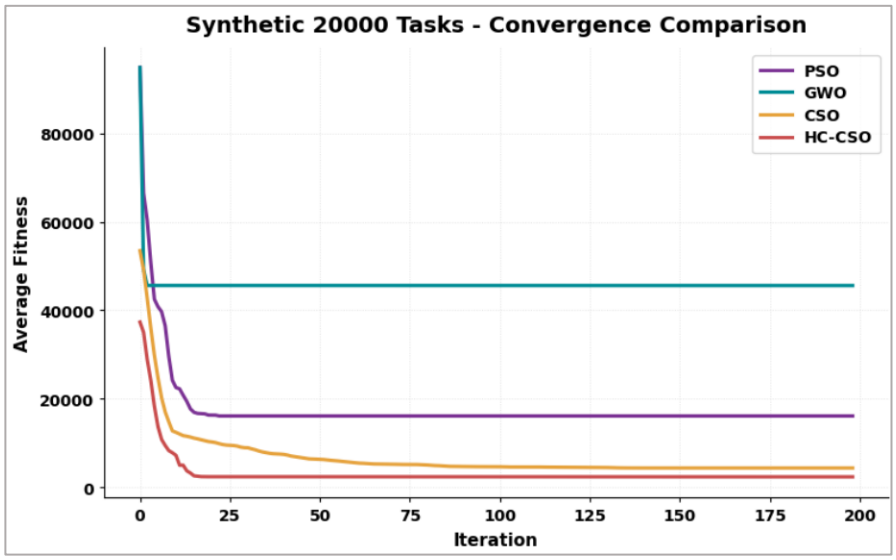


Fig. 9 Convergence Results for Synthetic 20000 Tasks

It achieves a 41.62% cost reduction compared to CSO, indicating that the HC-CSO is more economical than other algorithms. In case of DOI (Degree of Imbalance), HC-CSO achieves the lowest DOI of 1.69 i.e. it distribute the loads efficiently among all VMs compared to CSO, PSO and GWO algorithm. Figure 7 illustrates the optimization behavior of all four algorithms. HC-CSO converges rapidly during initial iterations and achieves lowest final average fitness. This indicates the efficient exploration and exploitation capacity of HC-CSO due to the integration of Crow Search and SMIW inertia weight.

For Synthetic 20000 big length tasks, HC-CSO achieves lowest makespan of 49294.97 Sec. followed by CSO (160827.11 Sec.), PSO (46230s7.01 Sec.) and GWO (909968.27 Sec.). HC-CSO reduces makespan by 69.35% compared to CSO algorithm and demonstrates its effectiveness in minimizing tasks execution time. In case of cost, HC-

DOI: 10.17148/IJARCCE.2021.101133

CSO records the lowest execution cost of 697685.13 compared to CSO (1180077.88), PSO (4599005.06) and GWO (15434864.89). It achieves a 40.88% cost reduction compared to CSO algorithm. It is indicating that the HC-CSO algorithm is more economical for utilization of cloud resources compared to other algorithms. In case of DOI, HC-CSO gets lowest DOI (Degree of Imbalance) that is 1.76. It is indicating that HC-CSO is capable to distribute the loads on available resources effectively. Figure 9 shows that HC-CSO algorithm converges rapidly during the initial iterations or generations and maintain the lowest average fitness for later stages. This is indicating, HC-CSO does not get trapped in local optima and explore more new solution in initial phases of generations or evolutions.

The experimental results from both scenarios demonstrate that the proposed HC-CSO algorithm can effectively solve complex cloud scheduling problems due to its balanced exploration and exploitation capabilities. This performance is achieved through the integration of the Crow Search and SMIW strategies. Another important observation is that HC-CSO shows improved performance in large scale data scheduling scenarios involving heterogeneous large lengths workloads particularly when using synthetic datasets with longer tasks lengths and larger number of tasks compared to other algorithms.

VI. CONCLUSION AND FUTURE SCOPE

This study presented a big data scheduling using HC-CSO swarm intelligence based scheduling algorithm. The HC-CSO algorithm is an integration of the Crow Search and SMIW strategies to improve exploration and exploitation capabilities. The algorithm was evaluated against CSO, PSO and GWO using real world scientific workflows namely CyberShake and Epigenomics along with large-scale synthetic workloads of 10,000 and 20,000 tasks. The experimental results demonstrate that HC-CSO algorithm consistently achieves better performance in terms of makespan, cost and degree of imbalance (DOI). Its convergence behaviour also indicates fast optimization and lower fitness values compared to competing algorithms. Particularly, the results for large-scale synthetic workloads demonstrate the effectiveness of HC-CSO in scheduling a large number of synthetic tasks at large scale cloud infrastructure i.e. 100 heterogeneous virtual machines in a datacenter.

The integration of Crow Search and SMIW strategies to improve exploration and exploitation help the algorithm to avoid premature convergence and identify better scheduling solutions. Overall, HC-CSO provides an effective approach for multi-objective cloud workload scheduling by balancing execution time, cost and load distribution. Future work will focus on extending the proposed approach towards energy-efficient scheduling and evaluating its performance in more complex and dynamic cloud environments. An extension of the algorithm can also be presented using other better exploration and exploitation techniques.

REFERENCES

[1]. A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. J. Epema, “Performance analysis of cloud computing services for many-tasks scientific computing,” IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 6, pp. 931–945, 2011.

[2]. S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, “Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service clouds,” Future Generation Computer Systems, vol. 29, no. 1, pp. 158–169, 2013.

[3]. H. Arabnejad and J. G. Barbosa, “List scheduling algorithm for heterogeneous systems by an optimistic cost table,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp. 682–694, 2014.

[4]. T. Ghafarian, and B. Javadi “Cloud-aware data intensive workflow scheduling on volunteer computing systems,” Future Generation Computer Systems, vol. 51, pp. 87–97, 2015.

[5]. M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, “Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds,” Future Generation Computer Systems, vol. 48, pp. 1–18, 2015.

[6]. X. Li, J. Xu, and Y. Yang, “A chaotic particle swarm optimization-based heuristic for market-oriented task-level scheduling in cloud workflow systems,” Computational Intelligence and Neuroscience, vol. 2015, Art. no. 718689, 2015.

[7]. N. Mansouri, B. M. H. Zade, and M. M. Javidi, “Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory,” Computers & Industrial Engineering, vol. 130, pp. 597–633, 2019.

[8]. P. Bryk, M. Malawski, G. Juve, and E. Deelman, “Storage-aware algorithms for scheduling of workflow ensembles in clouds,” Journal of Grid Computing, vol. 14, pp. 359–378, 2016.

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2021.101133

[9]. S. P. Praveen, K. T. Rao, and B. Janakiramaiah, "Effective allocation of resources and task scheduling in cloud environment using social group optimization," Arabian Journal for Science and Engineering, vol. 43, pp. 4265–4272, 2018.

[10]. Z.-G. Chen, Z.-H. Zhan, Y. Lin, Y.-J. Gong, T.-L. Gu, F. Zhao, H.-Q. Yuan, X. Chen, Q. Li, and J. Zhang, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," IEEE Transactions on Cybernetics, vol. 49, no. 8, pp. 2912–2926, 2019.

[11]. H. Hu, Z. Li, H. Hu, J. Chen, J. Ge, C. Li, and V. Chang, "Multi-objective scheduling for scientific workflow in multicloud environment," Journal of Network and Computer Applications, vol. 114, pp. 108–122, 2018.

[12]. J. Zhou, T. Wang, P. Cong, P. Lu, T. Wei, and M. Chen, "Cost and makespan-aware workflow scheduling in hybrid clouds," Journal of Systems Architecture, vol. 100, Art. no. 101631, 2019.

[13]. M. Kalra and S. Singh, "Multi-criteria workflow scheduling on clouds under deadline and budget constraints," Concurrency and Computation: Practice and Experience, vol. 31, no. 17, Art. no. e5193, 2019.

[14]. B. Kumar, M. Kalra, and P. Singh, "Discrete Binary Cat Swarm Optimization for Scheduling Workflow Applications in Cloud Systems," in Proc: IEEE International Conference on Computational Intelligence and Communication Technology (IEEE-CICT 2017), pp. 1-6, IEEE Xplore, 2017.

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

[16]. Y. Hu, H. Wang, and W. Ma, "Intelligent cloud workflow management and scheduling method for big data applications," Journal of Cloud Computing, vol. 9, Art. no. 39, 2020.

[17]. M. H. Shirvani, "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," Engineering Applications of Artificial Intelligence, vol. 90, Art. no. 103501, 2020.

[18]. A. Pasdar, Y. C. Lee, and K. Almi'ani, "Hybrid scheduling for scientific workflows on hybrid clouds," Computer Networks, vol. 181, Art. no. 107438, 2020.

[19]. S. Vakilinia, "Energy efficient temporal load aware resource allocation in cloud computing datacenters," Journal of Cloud Computing, vol. 7, Art. no. 2, 2018.

[20]. 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, Art. no. 2503153, 2017.

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

[22]. S. Chen, Z. Xu, Y. Tang, and S. Liu, "An Improved Particle Swarm Optimization Algorithm Based on Centroid and Exponential Inertia Weight," Mathematical Problems in Engineering, vol. 2014, pp. 1-14, 2014.

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

[24]. 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, vol. 7, no. 24, pp. 1-10, 2019.

[25]. H. Topcuoglu, H. Hariri, and M. Y. Wu, "Performance Effective and Low Complexity Task Scheduling for Heterogeneous Computing," IEEE Transactions on Parallel and Distributed Systems, vol. 13, no. 3, pp. 260-274 2002.

[26]. S. Chu, P. Tsai and J. Pan, "Cat Swam Optimization," In: Proc. of LNAI, vol. 4099, pp. 854-858, 2006.

AUTHORS BIOGRAPHIES



Jai Bhagwan is working as Asst. Professor in Computer Science & Engineering Department of Guru Jambheshwar University of Science & Technology, Hisar, India. He has more than 5 years of teaching experience. He received his MTech (CSE) degree from Maharishi Markandeshwar University, Mullana, India in 2011. Currently, he is pursuing his PhD (CSE) in the specialization of Cloud Computing and Swam Intelligence. His current research areas are Cloud Computing, Software Clone Detection and Swarm Intelligence.



Dr. Sanjeev Kumar is working as Professor in Computer Science & Engineering Department of Guru Jambheshwar University of Science & Technology, Hisar, India. He has more than 16 years of teaching experience. He is MTech (CSE) and PhD (CSE) in the area of Computer Networks. His current research areas are Cloud Computing, Computer Networks, Wireless Sensor Networks, Mobile Ad-hoc Networks etc.

 This work is licensed under a Creative Commons Attribution 4.0 International License