SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Crow Search Based PSO Algorithm for Load-Aware Workload Scheduling in Cloud Computing

Manoj[1], Jai Bhagwan[1*], Sanjeev Kumar[1], Sunila Godara[1], Seema Rani[2]

[1]Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India.
[2]State Institute of Engineering & Technology, Panchkula, Haryana, India.
*Corresponding Author: drjaicse@gmail.com, ORCID ID: 0000-0002-8708-4029

Abstract
Efficient task scheduling and load balancing are critical for improving performance in cloud computing environment containing heterogeneous VMs. This paper proposes an Improved Crow Search-Based Particle Swarm Optimization (ICPSO) algorithm that integrates SMIW inertia weight strategy with a Crow Search-based exploration mechanism to improve exploration-exploitation balance and reduce premature convergence of PSO algorithm. HEFT policy is used for common population initialization, while a combining fitness function considers makespan and degree of imbalance (DI). The proposed algorithm is evaluated using CyberShake, Montage, Inspiral and Sipht workflows each containing 1000 jobs with 50 heterogeneous VMs in CloudSim 3.0. The ICPSO algorithm is compared with IPSO, IJPSO and PSO in terms of makespan, DI, throughput and convergence. Experimental results show that the ICPSO algorithm consistently provides better performance across all workloads. Compared with IJPSO (second best algorithm), ICPSO achieves 11.49-36.70% improvement in makespan and 7.31-41.11% improvement in DI with higher throughput. The convergence results further demonstrate improved search capability of ICPSO and reduced premature convergence. These findings confirm the effectiveness of combining SMIW and Crow Search-based exploration for load-balanced cloud task scheduling.

Keywords: Cloud Computing, CRPSO, Degree of Imbalance (DI), Makespan, Particle Swarm Optimization, Virtual Machines (VMs).

1 Introduction

Cloud computing has adopted at wide level for delivering computational resource, storage and various services on demand through virtualized infrastructure. A cloud datacenter is generally consists of heterogeneous physical hosts and virtual machines with different processing capacities and memory configurations. As the number of diversity of user tasks are increased, the efficient task allocation to available VMs becomes one of the challenging optimization issues. Task or workload scheduling directly affects the important performance metrics such as makespan, response time, throughput, energy consumption etc. Particularly, load balancing is very much essential because improper task allocation can cause some VMs to become overloaded and some others remain underutilized. Which results an inefficient resource utilization and may increase execution time [1][2]. Task scheduling problem in cloud technology is mainly considered NP-hard problem because the number of possible task-to-VM mapping increases aggressively with the number of tasks and available resources. The conventional scheduling methods like FCFS, Round-Robin, Min-Min and Max-Min are simple to use but may not provide satisfactory solutions for large scale and heterogeneous computing environment. As a result, swam-intelligence based algorithms have received the attention of the scientists because these can explore the large search space effectively. These methods' population based searching mechanisms are especially suitable to obtain near optimal solution for task-resource allocations within reasonable computing time [3][4].

Particle Swarm Optimization is most widely investigate and adopted swarm intelligence techniques for cloud scheduling issues. Its particle-position and velocity-update mechanisms enable candidate schedules or solutions to learn from both individual and social-level experience. Various researches and theories have demonstrated that PSO-based scheduling can improve makespan, resource utilization while maintaining better load sharing among available VMs [5]-[8]. Particularly, load-balancing mechanisms integrated into PSO have been used to obtain overloaded and under-loaded VMs and redistribute the tasks toward the VMs having low load [5][6]. However, the

conventional or ordinary PSO may suffer from premature convergence and reduced diversity when we have large scheduling search space. Ant Colony Optimization is another kind of swarm-based approach which models the cooperative behaviour of ants pheromone-based search policy for cloud task scheduling and resource allocation. The adaptive pheromone methods have been proposed to improve the convergence to protect the ACO from stagnation. Multi-resource based approaches have considered the load balancing together makespan and cost based improved results [9][10]. Similarly, Artificial Bee Colony (ABC) based algorithms exploit the foraging behaviour of bees for searching improved task-resource allocations. ABC-based scheduling approach has also been used to balance the VMs workload and minimize the makespan. The hybrid approaches based on ABC method have further incorporated heuristic and reinforcement learning mechanisms for improving performance [11][12].
Other swarm-intelligence algorithms have also been demonstrated potential load balance-based scheduling in cloud technology. Firefly-based methods considered makespan, work-load simultaneously addressing the multi-objective nature of cloud scheduling [13]. Grey Wolf Optimization (GWO) has also been applied for cloud load balancing by considering VM load conditions [14]. Bird Swarm Optimization has similarly adopted for cloud load balancing with the objective of maintaining balance among various VMs for improving makespan and throughput [15]. Dragonfly, Grasshopper, and Fuzzy-ACO based methods have also demonstrated their capability for dynamic resource allocation and load balancing in cloud [16]-[18]. Therefore, effective load balance based task scheduling requires more than simply minimizing makespan. A robust scheduling technique should simultaneously maintain VM utilization to avoid under-loaded and overloaded situations while addressing makespan, cost and energy minimization. Recent research continuously favours hybrid and adaptive swarm intelligence-based strategies. Hybrid strategies combining exploration and exploitation, load-aware fitness, adaptive parameters mechanisms and local search methods can overcome the limitations of the conventional swarm-based algorithms [8][12][13]. This research direction provides a strong foundation for developing hybrid load-balanced task or workload scheduling algorithm capable of handling heterogeneous and large scale cloud environment.

The main contributions of this research paper are followings:
1. Modelling of Makespan and Degree of Imbalance.
2. Adopted SMIW inertia weight strategy for improving balance between exploration and exploitation.
3. Proposed Crow Search Based Mechanism to guide PSO to improve exploration i.e. more new solutions.
4. Comparison of newly designed ICPSO algorithm with other PSO family algorithms.

Rest of the paper is organized in five sections. Section 2 discusses literature review, section 3 represents system modeling and problem definition, section 4 explains proposed method, simulation results are represented in section 5 and finally conclusion and future direction is discussed in section 6.

2 Literature Review

Research on swarm intelligence-based load balancing in cloud computing has progressed from individual algorithms to adaptive and hybrid optimization approaches. One of the early researches by Babu and Krishna [1] proposed Honey Bee Behaviour inspired algorithm for load balancing to schedule independent non pre-emptive jobs among VMs. The method focused on achieving balanced VMs workloads and improving throughput and waiting time. Hashem et al. [2] proposed a Honey Bee based load balancing method that considered VM task count and processing time deviation to find appropriate resources. This method improved makespan, degree of imbalance, response time etc. Kumar and Prashar [3] designed a hybrid ACO and priority based ABC method. It demonstrated that the combining hybrid strategy can improve processing time, response time and cost while balancing the load. PSO algorithm has received particular attention. Alguliyev et al. [4] introduced αPSO-TBLB algorithm in which PSO was incorporated in load balancing strategy. The results evaluation reported improvements in makespan and resource utilization compared to conventional PSO. Pradhan and Bisoy [5] later designed a modified PSO algorithm for load balancing which especially targeted makespan, VM load distribution and resource utilization. Alsaidy et al. [6] addressed PSO limitation by initialization of population by LJFP and MCT heuristics. The results reported improvements in makespan, energy consumption and degree of imbalance. These researches indicates that both the search mechanism and quality of initial solutions are very much important for obtaining stable scheduling performance in cloud computing. ACO based techniques have also been widely adopted for load balancing. Liu [8] proposed an adaptive ACO scheduling method in which pheromone updating formula modified to reduce local stagnation. The resulting approach reduced cost and execution time while maintaining load balance. Muteeh et al. [9] developed a multi-resource load balancing using ACO for workflow scheduling in cloud. Their method considered makespan, cost and balanced resource utilization. Ragmani et al. [10] introduced a fuzzy based ACO technique for VM scheduling. The method demonstrated improved performance in cloud due to the integration of fuzzy technique. ABC methods have evolved from basic swarm optimization towards hybrid approaches for solving multi-objective scheduling. Kruekaew and Kimpan introduced a heuristic task scheduling based technique using

ABC for VM allocation and balancing of load. The results indicated improved makespan and load-balancing compared to ACO, PSO and improved PSO methods [11]. The same author later integrated ABC and Q-learning to design a mutli-objective load-balancing scheduler considering mult-objective scheduling [12]. These results suggest that hybridization with learning mechanism can improve the exploitation capabilities of conventional algorithms. Firefly and other swam based methods have demonstrated the scope of load balanced scheduling. Adhikar et al. [13] proposed a multi-objective Firefly-based workflow scheduling method that considered makespan, resource utilization etc. Abedi et al. [14] designed an improved Firefly-based dynamic resource allocation strategy in which load balancing, makespan, and migration rate where considered for optimization objectives. Sefati et al. [15] implemented GWO to cloud load balancing using VM reliability and load condition into optimization process. The response time and cost were improved. Mishra and Majhi [16] introduced a Binary Bird Swarm Optimization based load balancing approach where tasks and VMs were considered for scheduling and makespan along with resource utilization improved. Further study has investigated hybrid and problem specific swarm methods. Kiruthiga and Vennila [17] proposed a Grasshopper-based optimizer for load balancing. The designed approach used intuitionistic fuzzy clustering for improving resource scheduling. Latchoumi and Parthiban [18] introduced Quasi-Oppositional Dragonfly algorithm for load balancing in cloud technology. They considered execution time, cost and convergence behaviour to check the performance of the designed algorithm. Kaur and Mehta [19] applied an augmented Shuffled Frog Leaping algorithm for workflow scheduling in cloud and it demonstrated effective resource provisioning and scheduling. Finally, Manikndan et al. [20] introduced a hybrid Bee-mutated Whale Optimization algorithm to consider multi-objective based task scheduling. The hybrid method demonstrated good improvement in makespan, computational cost and resource utilization.

The literature demonstrates a clear transition from conventional single swarm based algorithm towards adaptive, hybrid and load-aware scheduling. However, various challenges are remaining including premature convergence, imbalanced exploration-exploitation, computational overhead etc. These limitations demand for a load-balanced scheduling algorithm that improves the makespan and other parameters. The most widely used and easy to implement PSO algorithm has been considered in this research. The limitations of poor exploration and premature convergence behaviour of PSO have been improved.

3 System Model and Problem Definition

This section is discussing problem formulation and system modelling.

3.1 Problem Formulation

Let $W = \{T_1, T_2, T_3, \dots, T_n\}$ be the workflow or collection of n tasks and $VM = \{VM_1, VM_2, VM_3, \dots, VM_m\}$ be the set of m virtual machines. The scheduler must control a mapping $f: F \rightarrow VM$ such that each task is assigned to one VM for execution. The example is displayed with 15 tasks and 5 VMs in Table 1. In case of workflows, the parent and child execution rules should not be violated. In this research, we have considered a datacenter having one host and 50 heterogeneous VMs for workload scheduling in cloud computing environment for simulation.

Table 1: Task-VM Allocation Representation

VM1	T1	T3	T7	T10	-	-
VM2	T4	T5	T9	T11	T15	-
VM3	T2	T6	T8	T12	T13	T14

3.2 Makespan Model

In cloud computing technology, the total completion time consumed by all tasks in a workflow or task group is called as makespan [21][22]. The mathematical model of the makespan is presented in Eq. (1).

$$\text{Makespan} = \max(F_i) \quad (1)$$

Where, i∈W, W is set or group of all tasks and F_i is finish time of tasks i.

3.3 DI Model

Degree of Imbalance (DI) is a method used to detect under-loaded and overloaded VMs by the optimization algorithm, which is discussed in Eq. (2).

$$DI = (Wd_{max} - Wd_{min})/Wd_{avg} \quad (2)$$

Where, $Wd_{max}, Wd_{min}, Wd_{avg}$ are maximum, minimum and average loads respectively among all VMs.

3.4 Throughput Model

In this research, we have considered throughput [21] metric to observe the proposed system's performance and it is calculated using Eq. (3).

$$\text{throughput} = \text{Total Tasks}/\text{Makespan} \quad (3)$$

3.5 Fitness Function Model

We have combined makespan and DI metrics in a single objective fitness function (F_x) which is calculated using Eq. (4). In the fitness function, we have considered number of datacenters and VMs to normalize the fitness values. In this research, the values of makespan and DI are minimized to check the fitness of algorithms.

$$F_x = \frac{w_1 \times \text{makespan} + w_2 \times \text{DI}}{\text{DC} \times \text{VMs}} \quad (4)$$

Where, DC is number of datacenters, VMs are virtual machines, w_1 and w_2 are the weights used to give the priorities to makespan and degree of imbalance. Here, slightly more priority is considered for makespan to make stable the system performance as end users priority is system's performance. The values of w_1 and w_2 are 0.6 and 0.4 respectively given that $w_1 + w_2 = 1$.

4 Methods and Workload Data

This section presents the method we have designed and implemented in this research.

4.1 PSO Theory

The particle swarm optimization algorithm has been introduced by J. Kennedy and R. Eberhart, in 1995 which is inspired by the birds flocking behaviour for food searching. This swarm based intelligence algorithm has applied various fields for optimization purposes. The mathematical modelling of PSO is discussed in Eqs. (5) and (6).

$$v_i^{t+1} = \omega v_i^t + c_1 rnd_1(pBest_i - x_i^t) + c_2 rnd_2(gBest - x_i^t) \quad (5)$$

Where, ω is the inertia weight to control the searching behaviour, $c_1$and c_2 are cognitive and social learning factors, rnd_1 and rnd_2 are random variables between (0, 1).

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (6)$$

Where, x_i is current position and v_i is current velocity.

4.2 Crow Search Algorithm Theory

The crow search algorithm designed by Askarzadeh (2016) is a population based intelligent searching algorithm [23]. This algorithm works on the behaviour of crow's intelligent searching for food. The basic idea is the crows remember its hiding food memory and may follow other crow too to steal its food. At a particle moment t, $crow_i$ decides to follow $crow_j$ to steal its food. The mathematical modelling of the CSA algorithm is given in Eq. (7).

$$X_i^{t+1} \begin{cases} X_i^t + r_i \times FL_i(M_j^t - X_i^t), & r_j \geq AP_j \\ Random\ Search & r_j < AP_j \end{cases} \quad (7)$$

Where, X_i^t is the current position, FL_i is the flight length (responsible for searching), a bigger flight length helps in global searching and lower flight length helps in local searching, r_i and r_j are random variables between (0, 1), M_j is the memory or best hiding location of $crow_j$.

4.3 SMIW Theory

We have adopted Self-Motivated Inertia Weight (SAIW) [22][24][25] which is a non-linear exponential-based decay strategy. This makes plays a major role to make the balance between exploration and exploitation, the same is suggested in [21]. The decay behaviour of the SMIW inertia strategy is represented in Fig. 1.

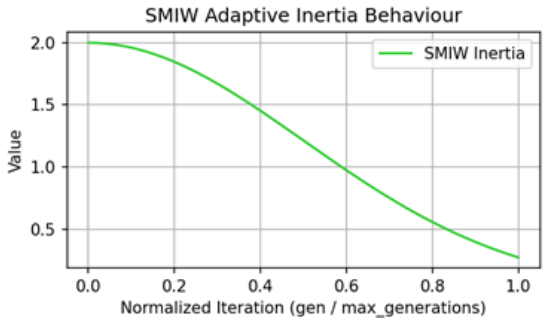

Figure 1: Inertia Weight Strategy Behaviour

The impact of the SMIW can be noted in the results and discussion section using algorithm convergence behaviour. The mathematical model of the SMIW is represented in Eq. (8).

$$\omega\ (itr) = \ \omega_{max} \times \exp\left(-\beta \times \left(\frac{itr}{itr_{max}}\right)^{\gamma}\right) \quad (8)$$

Where, $\omega(itr)$ is inertia weight at iteration or generation itr, ω_{max} represents maximum inertia weight (starting point) and β is a local search attractor which is set to and γ is global search attracter (both are set 2.0) in this research. This SMIW inertia weight has been integrated in PSO velocity update formula to make it more efficient and the algorithm is discussed in coming section.

4.4 HEFT Theory

The HEFT theory is used to maintain the DAG dependency and estimate the earliest finish time for workflows scheduling initialization, suggested in [22]. To decide tasks priorities, the HEFT algorithm [21][22] is capable to calculate the earliest finish time for both types of dependent and independent tasks as depicted in Algorithm 1. The HEFT algorithm is used for all experimental algorithms used in this research for common population generation for better comparison [22].

Algorithm 1: HEFT Algorithm

Input: Workloads T (Workflows or 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 to 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)
```

4.4 Proposed CRPSO Algorithm

In the proposed CRPSO algorithm [23], the SMIW inertia weight strategy is used in PSO's velocity update equation. The local crow search component movement is calculated and a hybrid velocity equation has been designed (see line 8) using PSO algorithm velocity and CSA movement. The hybrid velocity modelling is also depicted by Eq. 9.

$$v_i^d(t+1) = \alpha v_{PSO,i}^d(t) + (1-\alpha) v_{CSA,i}^d \quad (9)$$

Where, PSO component is derived from Eq. (5) and CSA component is derived from Eq. (7) excluding random search part [22]. The position update is calculated with original PSO's Eq. (6).

The working of the proposed ICPSO algorithm is discussed as:

1) At first, all parameters are initialized and the initial population is generated by HEFT (see Algorithm 1).
2) A random crow i.e. solution is generated from the population (solutions).
3) The hybrid velocity is calculated using Eq. (7) (see line 8 in Algorithm 2) and update the position.
4) The fitness is evaluated and stored in the memory.

Finally, this process (line 2-13 are repeated of Algorithm 2) continues till the condition satisfied (maximum generations are 200).

Algorithm 2: Proposed ICPSO Algorithm

Input: Generate Populations pop by HEFT policy, c_1, c_2, r_1, r_2, max_Generations etc.
Output: Best Schedule gBest

```
For Gen = 0 to max_Generations Do
    Calculate SMIW inertia weight ω using Eq. (8)
    For each particle Pi in pop Do
    Select a random crow_c from pop
        For each dimension d Do
            Update PSOVelocity using Eq. (5) // (use SMIW weight)
            //Calculate CSA local movement
            csaMove ← FL_local×rand()×(c.memory[d]-pi.position[d]
            //Calculate hybrid velocity
            Pi.velocity[d]← α×PSOVelocity+(1-α)×CSAMove // α is 0.8
            // Update position
            Pi.position[d] ← Pi.position[d]+Pi.velocity[d]
        End For
        Evaluate fitness(Pi)
        update pBest and gBest values
    End For
End For
return Best Schedule (gBest)
```

4.5 Workload Data

The workload data used in this research are CyberShake, Montage, Inspiral and Sipht each containing 1000 jobs. The structures of these scientific workflows are available online (https://pegasus.isi.edu).

5 Simulation Results and Discussion

CloudSim 3.0 simulator written in Java programming language has been used to conduct the experiments of this research. We have implemented all algorithms for a fair analysis and executed the algorithms' code using NetBeans 8.0 on a machine having Processor 12th Gen Intel® Core™ i5-1235U, RAM 16 GB, Storage 512 GB and Windows 11 OS. We have created 5 types of 50 heterogeneous VMs having computing power between 500-2500 MIPS, 512-2560 MB RAM and 1000 Mbps bandwidth. All algorithms are executed for 15 times to observe the stability. The rest of the simulation parameters details are displayed in Table 2. The performance parameters are makespan, Degree of Imbalance and throughput as described earlier.

Table 2: Simulation Parameters

Proposed ICPSO, IPSO and IJPSO Algorithms Properties	
Parameter	**Value**
Max Generations	200
No. of Cats (Population)	50
c1 and c2, r1 and r2, ω_{max}, β γ, α, (α in case of ICPSO, and IJPSO [21] only) and FL (FL in case of ICPSO only)	1.5 each, (0, 1) each, 2.0, 2.0, 2.0, 0.8 and 0.5 respectively
PSO Algorithm Properties	
Parameter	**Value**
Generations	200
No. of Particles	50
c_1, c_2, ω (in case of PSO only)	1.5, 1.5, 0.7

The results summary obtained after the experiments are shown in Table 3. The results are also described with the help of Figs. 2, 3, 4 for clear understanding of the impact of the proposed ICPSO algorithm. Table 4 depicts the improvement in makespan, DI and throughput of proposed ICPSO over IJPSO [21] algorithm.

Table 3: Summary of Makespan and Degree of Imbalance (DI)

Workload	Metric	Result Type	IPSO	IJPSO	ICPSO	PSO
CyberShake-1000	Makespan	Mean	1322.618	1312.350	830.708	2735.028
		Best	1135.014	1074.403	779.008	2315.305
		Worst	1594.984	1591.424	875.803	2977.555

	DI	Mean	5.469	5.387	3.245	39.752
		Best	3.385	3.745	2.625	21.494
		Worst	8.617	7.075	4.737	49.263
Montage-1000	Makespan	Mean	681.260	667.034	471.686	1120.944
		Best	570.871	557.574	425.165	845.993
		Worst	794.825	886.590	514.998	1339.128
	DI	Mean	5.347	5.135	3.024	32.242
		Best	4.703	3.343	2.321	16.559
		Worst	6.528	8.089	4.152	48.203
Inspiral-1000	Makespan	Mean	17717.959	16648.158	14106.336	27152.789
		Best	16895.339	15669.210	13625.601	23958.559
		Worst	19132.575	17804.739	14668.544	29890.967
	DI	Mean	6.083	5.283	3.300	31.431
		Best	4.305	3.948	2.601	18.606
		Worst	7.952	8.063	4.132	45.289
Sipht-1000	Makespan	Mean	18597.918	17805.053	15759.118	22902.666
		Best	16221.007	16223.981	13311.539	21921.956
		Worst	19859.516	20003.380	16730.709	23931.178
	DI	Mean	6.030	5.441	5.043	41.491
		Best	4.059	4.085	3.741	31.988
		Worst	8.330	7.152	6.840	49.286

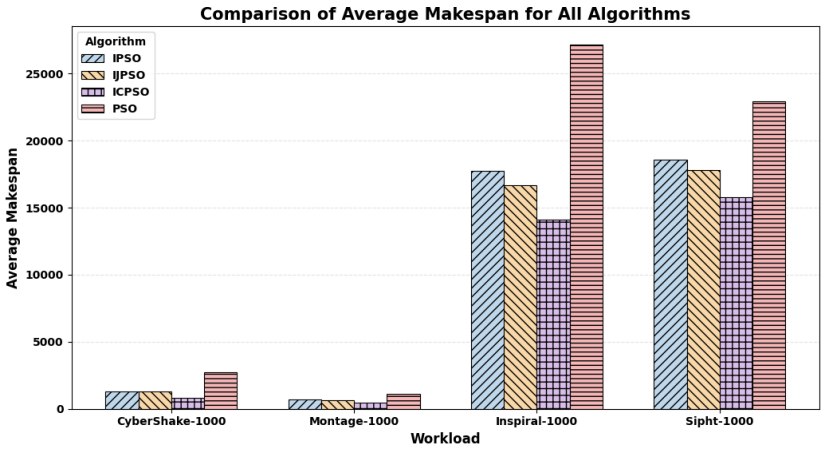

Figure 2: Makespan Comparison

In Fig. 2, PSO, IPSO, IJPSO [21] and proposed ICPSO algorithm have been compared. The ICPSO algorithm consistently performs better for all workloads. The PSO has the highest makespan across all four workloads. The makespan of ICPSO over IJPSO shows 11.49-36.70% improvement.

Figure 3 compares the average DI (degree of imbalance) for all algorithm across four workloads. The proposed ICPSO algorithm is indicating the best load balancing among all algorithms. PSO algorithm proves itself most imbalanced algorithm among its family. The improvement of of ICPSO over IJPSO is noted from 7.31-41.11% in terms of DI.

The proposed ICPSO algorithm improves the results due to good exploration produced by the local crow search mechnaism [22] and the SMIW inertia weight makes the good balance between exploration and exploitation. It supports good exploriation in early iterations and good exploitation in laters stages, the same has also been confirmed with the converge results shown in Fig. 5.

Figure 4 is indicating that the proposed algorithm is producing highest throughput among all algorithms such as PSO, IJPSO and IPSO.

Table 4: Percentage of Improvement of ICPSO over IJPSO

Workload	Makespan Improvement (%)	DI (%)	Throughput (%)
CyberShake-1000	36.70	39.76	9.77
Montage-1000	29.29	41.11	13.41
Inspiral-1000	15.27	37.54	18.10
Sipht-1000	11.49	7.31	13.05

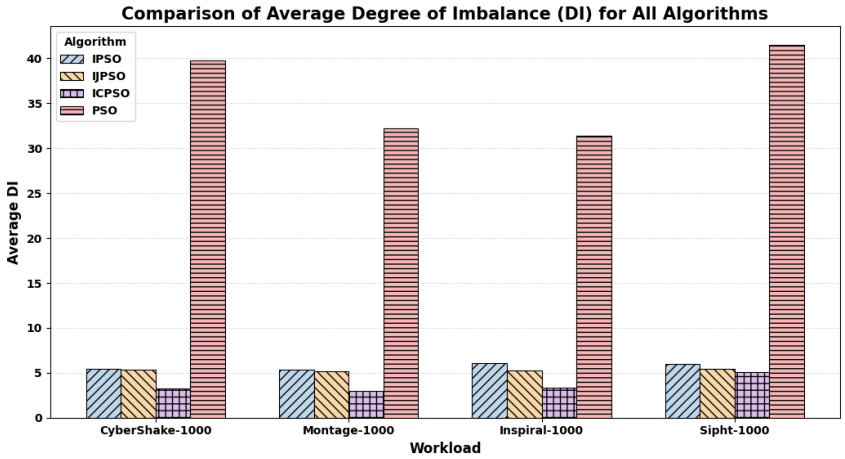

Figure 3: DI Consumption Comparison

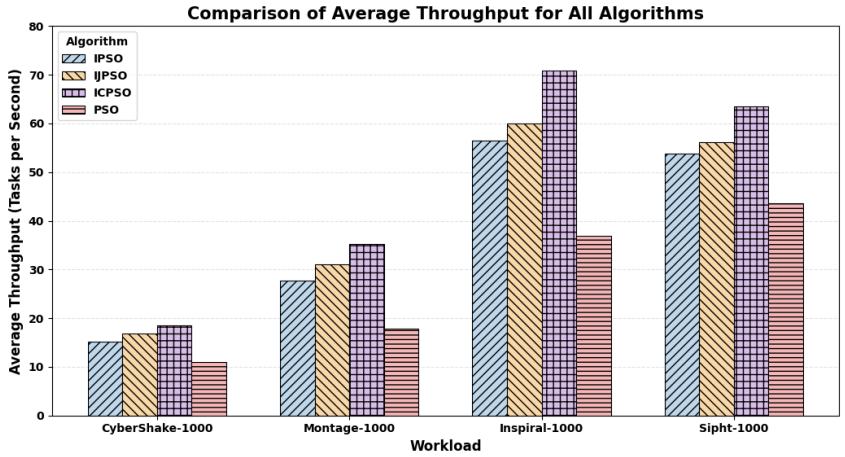

Figure 4: Throughput Comparison

In Fig. 5, the convergence behaviour of proposed ICPSO, IJPSO, IPSO and PSO algorithm has been presented. The proposed algorithm ICPSO (black line) initially decreased rapidly and later continues improving and reaching the lowest average fitness value for all four workloads containing 1000 tasks each. IJPSO and IPSO also showed rapid decrease in early stages but in later stages get stagnated compared to the proposed ICPSO. The PSO algorithm decreases more gradually and exhibited highest fitness value throughout the entire optimization process. The late-stage improvement of the ICPSO indicates that it continuously explores the search space after the convergence of other algorithms. Overall, the convergence behaviour of the proposed ICPSO algorithm demonstrate that this algorithm provides better solution quality and maintain search capability for longer avoiding premature convergence compared to other PSO family. This shows the stability and effectiveness of the proposed ICPSO algorithm.

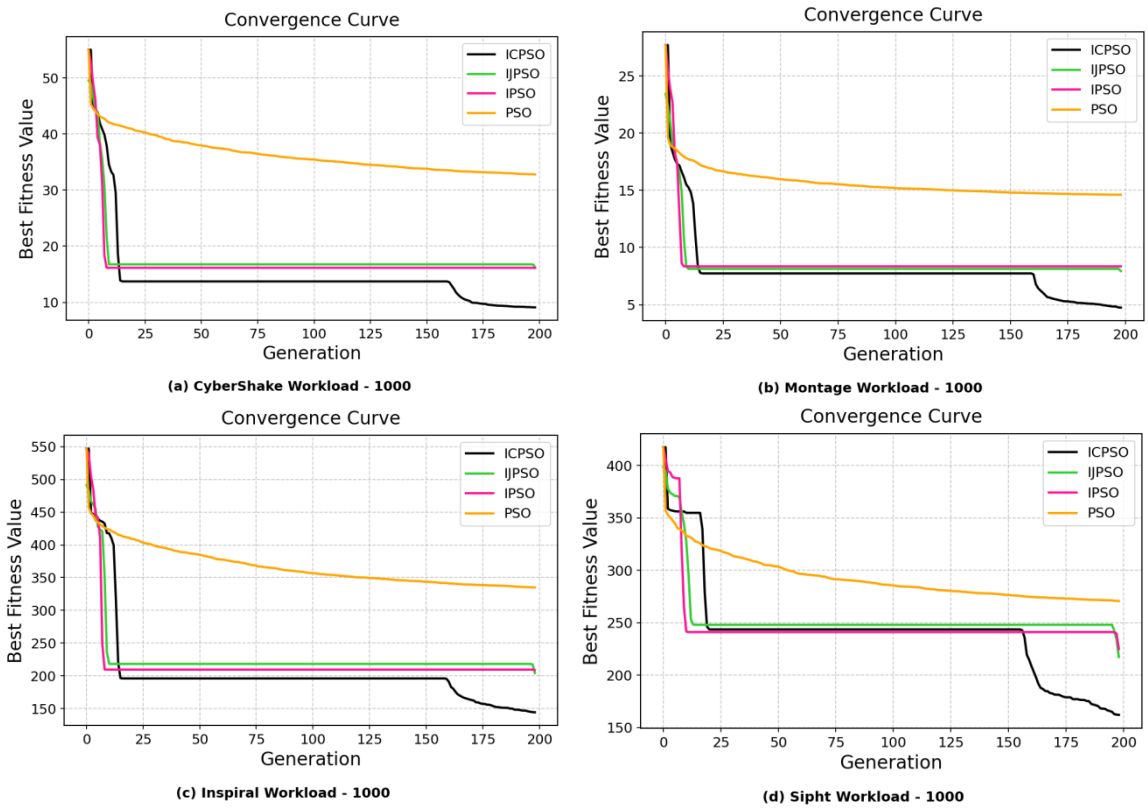

Figure 5: Convergence Results of All Algorithms

6 Conclusions and Future Direction

This paper proposed the ICPSO algorithm for load-balanced task scheduling in heterogeneous cloud environments. The integration of SMIW improves the exploration-exploitation balance while the Crow Search mechanism enhances exploration capability of PSO algorithm and avoids premature convergence. The experimental evaluation using four scientific workflows and 50 heterogeneous VMs demonstrated that the proposed ICPSO algorithm outperforms IJPSO, IPSO and PSO algorithms in account of makespan, degree of imbalance, throughput and convergence behaviour. Compared with IJPSO which is second best algorithm, the proposed ICPSO algorithm achieved 11.49-36.70% improvement in makespan and 7.31-41.11% improvement in DI. The results confirm that the proposed hybrid strategy can improve both scheduling performance and load balancing. Future work will extend ICPSO toward multi-objective scheduling by incorporating energy consumption, cost, resource utilization and other QoS parameters. Further, a deep learning based method can also be designed to improve the performance and dynamic task scheduling.

References

1. D. Babu L. D. and P. Venkata Krishna, “Honey bee behavior inspired load balancing of tasks in cloud computing environments,” Applied Soft Computing, vol. 13, no. 5, pp. 2292–2303, 2013.
2. W. Hashem, H. Nashaat, and R. Rizk, “Honey Bee Based Load Balancing in Cloud Computing,” KSII Transactions on Internet and Information Systems, vol. 11, no. 12, pp. 5694–5711, 2017.
3. R. Kumar and T. Prashar, “A bio-inspired hybrid algorithm for effective load balancing in cloud computing,” International Journal of Cloud Computing, vol. 5, no. 3, pp. 218–246, 2016.
4. R. M. Alguliyev, Y. N. Imamverdiyev, and F. J. Abdullayeva, “PSO-based Load Balancing Method in Cloud Computing,” Automatic Control and Computer Sciences, vol. 53, no. 1, pp. 45–55, 2019.
5. 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, vol. 30, no. 12, Article no. e4368, 2018.
6. A. Pradhan and S. K. Bisoy, “A novel load balancing technique for cloud computing platform based on PSO,” Journal of King Saud University–Computer and Information Sciences, vol. 34, no. 7, pp. 3988–3995, 2022.

7. S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, "Heuristic initialization of PSO task scheduling algorithm in cloud computing," Journal of King Saud University–Computer and Information Sciences, vol. 34, no. 6 part a, pp. 2370–2382, 2022.
8. H. Liu, "Research on cloud computing adaptive task scheduling based on ant colony algorithm," Optik, vol. 258, Article no. 168677, 2022.
9. A. Muteeh, M. Sardaraz, and M. Tahir, "MrLBA: Multi-resource load balancing algorithm for cloud computing using ant colony optimization," Cluster Computing, vol. 24, no. 4, pp. 3135–3145, 2021.
10. A. Ragmani, A. Elomri, N. Abghour, K. Moussaid, and M. Rida, "FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing," Journal of Ambient Intelligence and Humanized Computing, vol. 11, no. 10, pp. 3975–3987, 2020.
11. B. Kruekaew and W. Kimpan, "Enhancing of Artificial Bee Colony Algorithm for Virtual Machine Scheduling and Load Balancing Problem in Cloud Computing," International Journal of Computational Intelligence Systems, vol. 13, no. 1, pp. 496–510, 2020.
12. B. Kruekaew and W. Kimpan, "Multi-Objective Task Scheduling Optimization for Load Balancing in Cloud Computing Environment Using Hybrid Artificial Bee Colony Algorithm With Reinforcement Learning," IEEE Access, vol. 10, pp. 17803–17818, 2022.
13. M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach," Applied Soft Computing, vol. 93, Article no. 106411, 2020.
14. S. Abedi, M. Ghobaei-Arani, E. Khorami, and M. Mojarad, "Dynamic Resource Allocation Using Improved Firefly Optimization Algorithm in Cloud Environment," Applied Artificial Intelligence, vol. 36, no. 1, pp. 1–27, 2022.
15. S. Sefati, M. Mousavinasab, and R. Z. Farkhady, "Load balancing in cloud computing environment using the Grey wolf optimization algorithm based on the reliability: performance evaluation," The Journal of Supercomputing, vol. 78, no. 1, pp. 18–42, 2022.
16. K. Mishra and S. K. Majhi, "A binary Bird Swarm Optimization based load balancing algorithm for cloud computing environment," Open Computer Science, vol. 11, pp. 146–160, 2021.
17. G. Kiruthiga and S. Mary Vennila, "Robust resource scheduling with optimized load balancing using grasshopper behavior empowered intuitionistic fuzzy clustering in cloud paradigm," International Journal of Computer Networks and Applications, vol. 7, no. 5, pp. 137–145, 2020.
18. T. P. Latchoumi and L. Parthiban, "Quasi Oppositional Dragonfly Algorithm for Load Balancing in Cloud Computing Environment," Wireless Personal Communications, vol. 122, no. 3, pp. 2639–2656, 2022.
19. P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm," Journal of Parallel and Distributed Computing, vol. 101, pp. 41–50, 2017.
20. N. Manikandan, N. Gobalakrishnan, and K. Pradeep, "Bee optimization based random double adaptive whale optimization model for task scheduling in cloud computing environment," Computer Communications, vol. 187, pp. 35–44, 2022.
21. J. Bhagwan, S. Rani, S. Godara et al., "IJPSO: an improved hybrid PSO algorithm for Multi-Type and Large Scale Data Scheduling in Cloud Computing," International Journal of Artificial Intelligence and Machine Learning, vol. 6, no. 3s, pp. 399-412, 2026.
22. J. Bhagwan, and 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.
23. A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm, Computer & Structures, vol. 169, pp. 1-12, 2016.
24. S. Chen, Z. Xu, and S. Liu, "An improved particle swarm optimization algorithm based on centroid and exponential inertia weight," Mathematical Problems in Engineering, vol. 2014, Article no. 976486, 2014.
25. T. O. Ting, Y. Shi, S. Cheng, and S. Lee, "Exponential inertia weight for particle swarm optimization," Lecture Notes in Computer Science, pp. 83-90, 2012.