SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Adaptive PSO Algorithm for Resource Allocation to Large Scale Data in Heterogeneous Cloud Computing Environment

Abhishek Bishnoi[1], Jai Bhagwan[1]*, Sanjeev Kumar[1]

Department of Computer Science & Engineering, Guru Jambheshwar University of Science & Technology, Hisar, Haryana, India

*Corresponding author: drjaicse@gmail.com

Abstract

The resource management plays a key role in cloud computing environment to minimize the load balancing issue. In the era of artificial intelligence, a large number of workflows are generated to process on cloud servers. Efficient resource allocation in cloud computing users' incoming tasks is one of the major issues. Efficient resource management not only improves overall system performance but reduces computing cost, degree of imbalance etc. Therefore in this paper, we have proposed an efficient resource allocation algorithm namely APSO. The proposed APSO algorithm is designed with the help of the SMIW inertia weight i.e. exponential decay factor and adaptive c1 and c2 coefficients. The adaptive social and cognitive learning factors improve the exploration and exploitations whereas the inertia weight makes balance between exploration and exploitations to find best solutions. To prove the capability of the proposed APSO algorithm, the experiments have been conducted 20 times. Based on these experiments, the proposed APSO reduced 14% makespan in case of CyberShake workflow, 14% in case of Montage workflow, 10% in case of Inspiral workflow, and 11% in case of Epigenomics over its competitor IPSO. In case of cost, the proposed APSO reduced 49% cost in case of CyberShake workflow, 53% in case of Montage workflow, 29% in case of Inspiral workflow, and 10% in case of Epigenomics over its competitor IPSO. In case of degree of imbalance (DI), the proposed APSO reduced 27% DI in case of CyberShake workflow, 31% in case of Montage workflow, 12% in case of Inspiral workflow, and comparable in case of Epigenomics over its competitor IPSO. Also, the proposed APSO improved higher throughput in all cases.

Keywords: Cloud Computing, HEFT policy, Particle Swarm Optimization (PSO), Large Data Scheduling, Virtual Machines (VMs)

1. Introduction

Cloud computing has become one of the most influential paradigms in modern distributed computing by providing on-demand access to computational resources. The demand of efficient resource management in cloud computing has rapidly increased due to rapid growth of cloud applications in the area of big data, scientific workflows, IoT, and artificial intelligence etc. In cloud computing, users submit a large scale tasks that must be scheduled to virtual machines. Therefore, task scheduling or VM allocation have gained more attention that directly influence system performance and quality of service (Qos) [1]. Task scheduling is an NP-hard optimization problem because the scheduler must regulate an efficient mapping of tasks to VMs. Traditional algorithms like FCFS, Round Robin, Max-Min, and Min-Min are simple but often fail to provide optimal solutions for large-scale and dynamic environment [2]. So, the scientists have progressively adopted swarm intelligence algorithms inspired by nature. Among these algorithms, Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Cat Swarm Optimization (CSO) and Grey Wolf Optimization (GWO) have gained momentous attention to enhance scheduling efficiency. ACO adopts the foraging behaviour of ants and utilizes the pheromone trails to discover near optimal task-resource mappings [3]. The ACO algorithm has demonstrated a strong exploration capability and efficient load distribution in cloud computing. However, this algorithm suffers from slow and premature convergence when working with large search space [4]. Particle Swarm Optimization is another widely used algorithm due to its simplicity and global searching capabilities. Nevertheless, the standard PSO experiences premature convergence during later stages. This issue may affect finding better solution in large search space [5][6]. The GWO algorithm developed in 2014 by Mrijalili et al., works based on the leadership hierarchy and hunting mechanism of grey wolves. The GWO has also applied to solve various cloud computing

problems like scheduling and resource allocation problems. Despite the effective of GWO, it can exhibit slow local search near the global optimum region and can trapped in local search [7][8].

Scientists have developed various hybrid algorithms to overcome various limitations of individual algorithms; these hybrid algorithms combine various capabilities of multiple meta-heuristic methods. Various hybrids ACO-PSO, ACO-GWO, PSO-GWO and other methods have been developed and achieve significant improvements in makespan, computation cost, throughput and resource utilization [9-11]. Furthermore, various other strategies such as adaptive parameter controlling, chaotic maps and machine learning-assisted methods have also been incorporated to enhance task scheduling performance [12-14]. A significant progress has been achieved so far still several challenges remain unresolved. Most recent studies mainly focus only on makespan while ignoring load balancing, SLA satisfaction, and resource utilization. Many researches have ignored the dynamic nature task scheduling objectives in modern cloud computing [15]. Moreover, many algorithms have been developed to solve other areas optimization problems. These limitations attract the scientists to develop more hybrid and adaptive task scheduling algorithms.

The main contributions of this research papers are:

- Considered Workflows Scheduling due to the significant improvement in emerging Internet of Things (IoT) technology which generates workflows for execution at various levels such as edge, fog and cloud computing.
- Introduced HEFT algorithm to maintain the child-parent dependencies and to generate ranks for parent and child tasks to get executed.
- Introduced SMIW inertia weight and adaptive cognitive and social learning c1 and c2 coefficients. SMIW weight control is used3 for making balance between exploration and exploitation and adaptive c1 and c2 are introduced to enhance local and global searching during learning process.
- Comparison of proposed APSO algorithm with PSO, IPSO and GWO algorithm.

The remaining paper is structured as: the literature review is explained in section 2, section 3 discusses the problem definition and modelling, proposed algorithm and adaptive strategies have been discussed in section 4, section 5 discusses simulation setup and results discussion and finally section 6 describes the conclusion and future scope of this research.

2. Related Work

Cloud task scheduling or resource allocation put its direct impact on cloud service performance. So, it attracts the scientists to explore various nature-inspired optimization techniques to address NP-hard problem of task scheduling. Tawfeek et al. [3] proposed ACO-based scheduling techniques for cloud computing environment. The method used pheromone trails to guide the task-VMs assignment decisions and achieved lower makespance compared to FCFS and Round-Robin scheduling method. Following this work, Li et al. [4] enhanced the standard ACO method by injecting load balancing considerations. The results related to resource utilization and execution delay proved the efficacy of the proposed algorithm. Rajakumari et al. [12] used fuzzy logic with ant colony optimization and addressed better quality of service metrics. Similarly, Chandrashekar et al. [13] introduced a hybrid weighted ant colony optimization (HWACSO) algorithm in order to improve convergence speed. Adaptive heuristic information and weighted methods improved the results.

Particle Swarm Optimization has also been applied to cloud scheduling problem extensively. Pandey et al. [5] proposed PSO based workflows scheduling method that addresses significance in execution cost reduction compared to traditional algorithms. Verma and Kaushal [6] proposed improved PSO to introduce deadline aware scheduling method for workflows applications. The proposed approach achieved improved cost-performance trade-offs under various constrained environments. Liu et al. [9] designed a PSO-ACO scheduling method in cloud computing. PSO method was used for global exploration whereas ACO refined the local solutions. The proposed method found better in terms of makespan, cost and resource utilization. The similar findings were reported by Liu et al. [10], who introduced ACO-PSO based method for improving solution quality and convergence speed. Kumar et al. [8] applied GWO to cloud task scheduling problem and the better results were reported better makespan and cost compared to PSO and Flower Pollination Algorithm for multiple scenarios. Singh et al. [11] proposed a load balancing algorithm using the capabilities of GWO and PSO. GWO algorithm enhanced the exploration capabilities and PSO increased exploitation capability of the proposed approach. This hybrid algorithm produced better results in terms of convergence speed and load balancing. For dynamic workload adjustment a reinforcement learning based GWO was developed in [14].

Gupta et al. [16] proposed an energy-efficient task scheduling framework which is based on Particle Swarm Optimization for cloud environments. The research considered execution time and energy consumption. The

proposed approach found better compared to traditional PSO based strategy. Zhou et al. [17] proposed PSOMCD algorithm for load balancing using PSO and modified crowding distance. The proposed method provided better makespan, energy utilization with refined throughput. Madni et al. [18] presented a survey on workflows scheduling techniques and suggested the weakness of ACO, GA and PSO. Kumar and Venkatesan [19] designed a hybrid GWO and Ant Colony Optimization algorithm for tasks assignment in cloud computing environment. The ACO improved the local searching whereas GWO improved global searching. Experimental analysis revealed that the hybrid algorithm improved throughput along with makespan and resource utilization.

Despite these advancements, the existing researches have several limitations. Most algorithms did not considered QoS metrics such as makespan, cost, degree of imbalance and throughput for workflows scheduling. A single algorithm is not capable to produce optimized workflows scheduling in cloud computing. The Particle Swarm Optimization algorithm is easy to implement and has good convergence than ACO, Max-Min, and others [27][28]. However, it gets stagnated in local optima in later stages of learning for finding solution. Also, it has the lack of balance between local and global searching. To overcoming this issue, a hybrid PSO scheduling algorithm is requirement for large scale workflows scheduling to optimize Quality of Service metrics.

3. Problem Definition and Modelling

In this research, the resource allocation (VMs allocation to Tasks) is considered. We have considered 50 virtual machines in a host machines for workflows scheduling. A workflow is a group of tasks (directed acyclic graph) which are connected to each other in a parent and child relations. A datacenter containing several host machines provide the IaaS infrastructure to the users in a pay-per-use model. The computation service especially scheduling is provided in the form of virtualization i.e. virtual machines (VMs) inside the hosts. Let's say there are n number of tasks and m number of VMs. The example of tasks-VMs assignment problem is represented in Table 1. T1, T2 …… T17 represents the tasks and V1, V2, V3 represents the virtual machines.

Table 1. Tasks-VMs Assignment Representation

Tasks-VMs Mapping						
VM1	T1	T3	T7	T10	T15	T16
VM2	T4	T5	T9	T11	T17	-
VM3	T2	T6	T8	T12	T13	T14

3.1 Makespan Modelling

Makespan is known as the maximum schedule length i.e. completion time taken by an algorithm to complete n number of tasks, which is computing using Eq. (1).

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

Where, i∈T, T is set of all tasks and F_i is the finish time of tasks i.

3.2 Cost Modelling

The service provider of cloud services offers the services on pay-per-use model. Keeping this pay-per-use model in mind, we have designed a cost model inspired by [21-23]. It is assumed that a datacenter contains one host and several VMs, consume the computation cost as given in Eq. (2).

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

The Migration factor (MF) is given in Eq. (3). Let's say: HT is number of host machines, DC is number of data-centers, Migs is number of tasks migrations and VM_u is number of VMs used, then:

$$MF = \frac{DC}{HT} \times \frac{Migs}{VM_u} \tag{3}$$

The computation cost factor is given in Eq. (4). Let's say: V is a set of VMs, PT_j is processing time on VM_j, $Memory_j$ is total task memory on VM_j, $MIPS_j$ is MIPS of VM_j and $DC_{TotMips}$ is datacenters' MIPS, then:

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

Here, VM scaling factor is used to scale up the cost if the number of VMs is increased using pay-per-usage model, and it is calculated using Eq. (5). Let's say: w_{scale} is VM scaling weight, then:

$$SF = 1 + w_{scale} \times V^2 \quad (5)$$

Final CF (cost factor) is calculated using Eq. (6). Let: norm is a normalization factor i.e. 1000 to balance the cost for realistic environment.

$$CF = norm \times CF_{base} \times SF \quad (6)$$

3.3 DI Modelling

In cloud task scheduling or VM allocation, the degree of imbalance (DI) is metric used to measure how workloads are distributed on VMs. It is computed using Eq. (7). In this research, it has been included in the fitness function to manage the VMs load.

$$DI = (T_{max} - T_{min}) \,/\, T_{avg} \quad (7)$$

Where, T_{max} is maximum load among all VMs, T_{min} is minimum load among all VMs and T_{avg} is average load of all VMs.

3.4 Throughput Modelling

Throughput [26] is a performance metric to observe the system's performance; it is calculated using Eq. (8). The throughput metric has not been included in the fitness function to avoid complex fitness calculation for a scheduler.

$$throughput = Total\ Tasks/Makespan \quad (8)$$

3.5 Fitness Function

To optimize the makespan, cost and degree of imbalance (DI), we have combined these in a single objective fitness function (F_x) which is reflected in Eq. (9). We have used number of datacenter (DC) and VMs to normalize the fitness values. We are minimizing the values of fitness function during the number of generations to optimize the makespan, cost and DI.

$$F_x = \frac{w_1 \times makespan + w_2 \times cost + w_3 \times DI}{DC \times VMs} \quad (9)$$

Where w_1, w_2 and w_3 are the weights used to give the priorities to makespan, cost and DI. We are giving slightly more priority to makespan and cost to stable the system performance under minimum budget along with efficient load balancing. The values of w_1, w_2 and w_3 are 0.5, 0.3 and 0.2 respectively given that $w_1 + w_2 + w_3 = 1$.

4. Proposed Methods Theory

This section is explaining the proposed framework in detail. The coming sub-sections are describing HEFT algorithm theory, PSO theory, SMIW inertia weight, adaptive coefficients theory and proposed APSO algorithm.

4.1 HEFT Algorithm Theory

The HEFT algorithm [22] is used to rank the workflows task based on the average of earliest finish time and tasks successors as displayed in Algorithm 1. If the system founds workflows than the ranking or priorities are set based on earliest finish time for parents and children tasks. In case the tasks are not dependent only the average is used. The HEFT is used to initialize the VMs only in this research and the initial generated by HEFT is fed in the proposed APSO algorithm.

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 = 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)
```

4.2 PSO Theory

The particle swarm optimization algorithm was designed by J. Kennedy and R. Eberhart in 1995 to solve real world problems. The algorithm is based on the flocking birds' behaviour and it can be better understood using its mathematical model given in Eq. (10) and (11).
The mathematical model to update its velocity is given in Eq. (10).

$$v_i^{t+1} = \omega v_i^t + c_1 r_1 (pBest_i - x_i^t) + c_2 r_2 (gBest - x_i^t) \quad (10)$$

The mathematical model for updating the positions to find out new solution can be represented by Eq. (11).

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (11)$$

Where, ω is inertia weight and it came into existence later developments not in original PSO, c_1 is cognitive coefficient, c_2 is social coefficient, r_1 and r_2 are random numbers in the range (0, 1), x_i is current solution's position and v_i is current velocity.

4.3 SMIW Exponential Decay Inertia Weight

The studies [22-25] suggest using exponential decay-based inertia weight to make the balance between exploration and exploitation for algorithm efficiency. The mathematical model of the Self-Motivated Inertia Weight (self-adjustable) is given in Eq. (12).

$$\omega = \omega_{max} \times \exp\left(-l \times \left(\frac{itr}{itr_{max}}\right)^g\right) \quad (12)$$

Where, ω_{max} is the initial inertia weight set to 2.0. l is the local searching attractor and g is the global searching attractor. The local and global searching attracters are used to make the balance among the local and global searching. Basically g controls the decay rate and larger value of g *than* l guides the algorithm for exploitation and smaller value guides for exploration. Here, ω decreases with algorithm's evolution during generations increments. The exponential is defined as decay function. l and g are set to 2.0 to make fine focus on balancing the searching. In early generations the algorithm moves toward exploration whereas in later stages it moves toward exploitation.

4.4 Adaptive Cognitive and Social Coefficients Theory

The theory of adaptive cognitive and social coefficients c1 and c2 is given in [20] in 2004. The authors suggested these two parameters are considered very important to find optimum solution efficiently. A higher value of c1 compared with c2 will cause unnecessary movement of the candidate solutions in search space. In contract, higher value of c2 may lead to premature convergence. So, it is suggested to make tuning between these two. It is necessary to encourage the candidate solutions to move in entire search space for the efficiency of the algorithm.
So, in this research both c1 and c2 acceleration coefficients are introduced for task scheduling. The mathematical models of these two are given in Eq. (13) and (14).

$$c_1(t) = c_1^{max} - (c_1^{max} - c_1^{min}) \frac{t}{T_{max}} \quad (13)$$

$$c_2(t) = c_2^{min} - (c_2^{max} - c_2^{min}) \frac{t}{T_{max}} \quad (14)$$

Where, $c_1(t)$ is cognitive learning factor, $c_2(t)$ is social learning factor, t denotes the current generation or iteration and T_{max} represents the maximum number of generations.
In this research, c_1 is decreased linearly from 2.5 to 0.5 and c_2 is increased linearly from 0.5 to 2.5. This adaptive strategy exercises exploration during the early search stages and exploitation during later stages. This strategy

improves the convergence behaviour of the proposed APSO algorithm. The proposed APSO algorithm is discussed in next section.

4.5 Proposed APSO Algorithm

The proposed APSO algorithm is a combination of the inertia weight and the adaptive cognitive and social learning factors theory given in section 4.4. Now using Eq. (12) - (14), the new velocity update formula becomes as given in Eq. (15).

$$v_i^{t+1} = \omega v_i^t + c_1(t) r_1 (pBest_i - x_i^t) + c_2(t) r_2 (gBest - x_i^t) \quad (15)$$

The position update will remain same as given in Eq. (10).

The newly APSO algorithm is represented in Algorithm 2. The initial solutions are fed into APSO then inertia weight, $c_1(t)$ and $c_1(t)$ are computed. In line 7, the velocity of the proposed APSO algorithm is updated using Eq. (15). In next step (line 8), the position is updated using Eq. (10); this process continues for all dimensions i.e. number of tasks. After the completion of this step, the fitness of the solution (tasks-VMs assignment) is evaluation. The pBest and gBest values are updated. This process continues until the max_generations are achieved. Finally, the best solution i.e. schedule is returned.

Algorithm 2: Proposed APSO Algorithm

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

1. **For** Gen = 0 to max_Generations **Do**
2. Compute SMIW inertia weight ω using Eq. (12)
3. Compute $c_1(t)$ using Eq. (13)
4. Compute $c_2(t)$ using Eq. (14)
5. **For each** particle in pop **Do**
6. **For each** dimension d **Do**
7. Update velocity using Eq. (15) // using SMIW, $c_1(t)$ and $c_2(t)$
8. Update Position using Eq. (10)
9. **End For**
10. Evaluate fitness
11. update pBest and gBest values
12. **End For**
13. **End For**
14. return Best Schedule (gBest)

5. Simulation Setup and Results Discussion

This section presents the results obtained after experiments using the proposed APSO, IPSO, PSO, and GWO algorithm. The experiments were conducing using CyberShake, Montage, Epigenomics and Inspiral datasets (the structures of these datasets are available on https://pegasus.isi.edu/workflow_gallery/ website). The simulation is performed in CloudSim 3.0 tool which is written in Java programming language. The experiments were conducted large scale tasks. For this research, 50 heterogeneous virtual machines have been used. The characteristics of VMs and algorithms' parameters are displayed in Table 1

Table 2. Simulation Parameters

Parameter	Value
No. of VMs	50
VMs RAM	512-2556 MB
Computing Power	500-2500 MIPS
Bandwidth	1000 Mbps
PEs (CPUs)	1 For each VM
PSO, IPSO and Proposed APSO Algorithms Properties	
No. of Population	100
No. of Generations	250
C_1, C_2, r_1 and r_2	1.5, 1.5, r_1 and r_2 random in (0, 1)

GWO	
No. of Population	100
No. of Generations	250

5.1 Results Analysis

All algorithms have been run for 20 times to check the stability and reliability and average results of makespan, cost, DI are described in this section along with overall throughput.

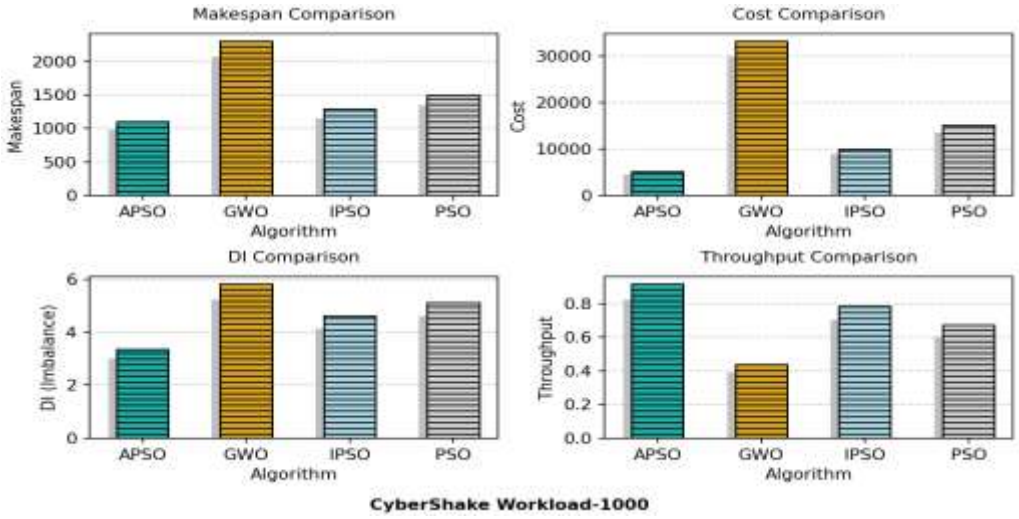

Figure 1. CyberShake Workload Results

The experimental results on CyberShake workload containing 1000 tasks demonstrate that the proposed APSO algorithm outperform IPSO, PSO and GWO algorithm. Our APSO algorithm achieves minimum makespan, cost, and degree of imbalance (DI) while achieving higher throughput compared to all algorithms as shown in Fig. 1. The adaptive search mechanism of our proposed APSO algorithm effectively balances exploration and exploitation. It leads fast task completion and improved tasks-VMs mappings.

The results from Fig. 2 demonstrate that the proposed APSO again outperforms IPSO, PSO and GWO algorithm in terms of makespan, cost, degree of imbalance and throughput. The lower makespan achieved by proposed APSO algorithm is indicating faster completion of the Montage workflow. IPSO and PSO also work well but remain inferior to APSO. The GWO performs worst in this scenario.

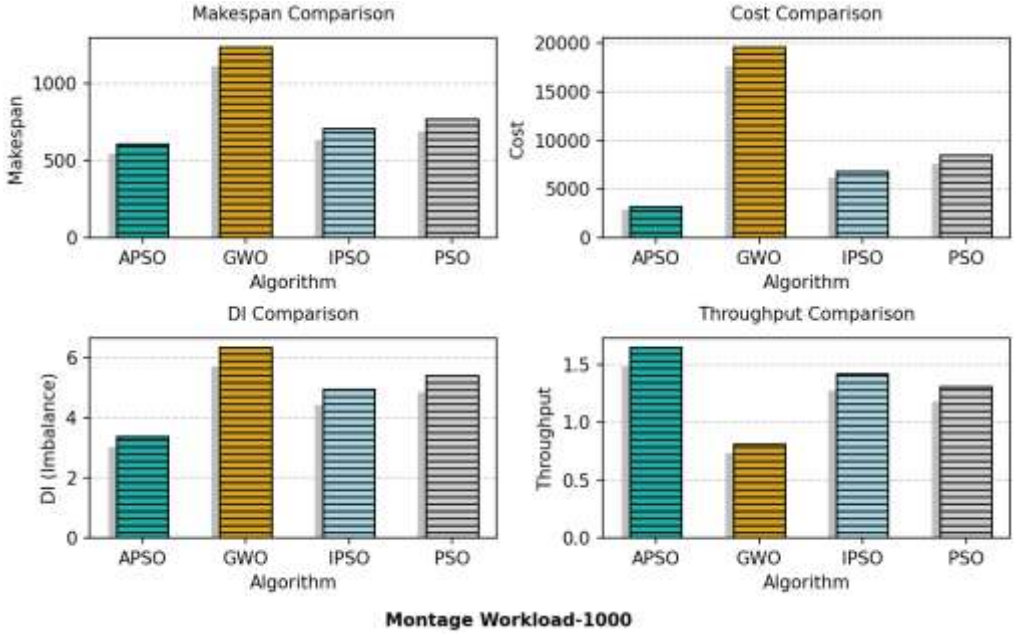

Figure 2. Montage Workload Results

The adaptive technique helps the proposed method to find better VM-tasks assignments by balancing the exploration and exploitation. The algorithm detects efficient VMs by considering fitness values and then assigns tasks efficiently by ignoring weaker VMs. The degree of imbalance parameter used in the fitness function helps the proposed algorithm to detect under-loaded and overloaded VMs for better scheduling. The montage dataset contains various shorts tasks so the makespan and cost are lower compared to CyberShake data.

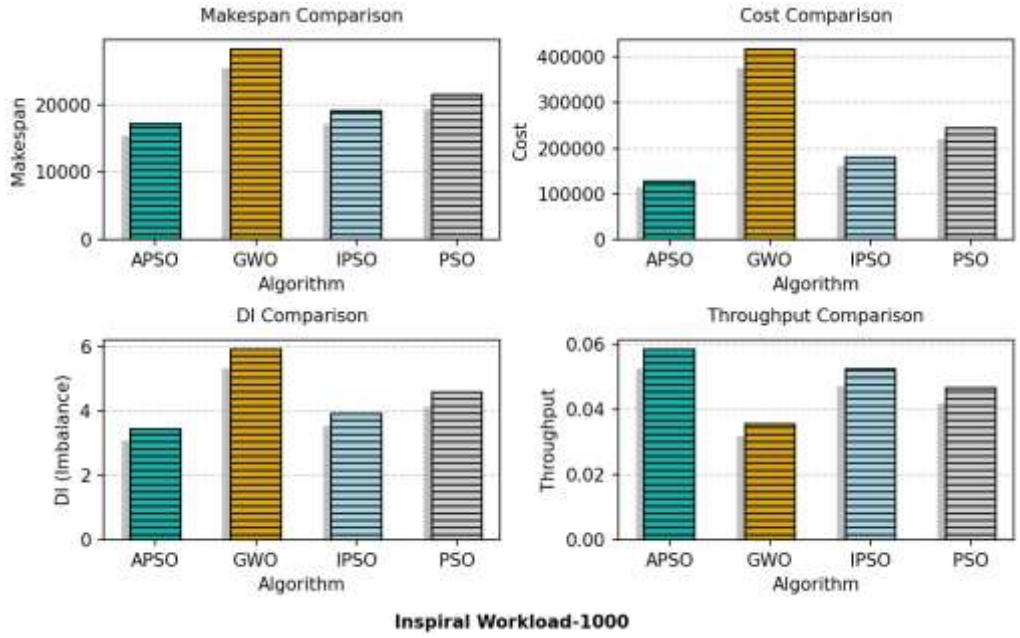

Figure 3. Inspiral Workload Results

In Fig. 3, the experimental results evaluation on Inspiral dataset is presented. The results demonstrate the superiority of our proposed APSO algorithm over PSO, IPSO and GWO in a large scale cloud scheduling environment considering 1000 tasks. The degree of imbalance, makespan, cost and throughput prove that the IPSO got the second position while the proposed APSO gained higher performance due to its adaptive nature. The adaptive method searches the solutions efficient in local and global regions carefully using fitness function.

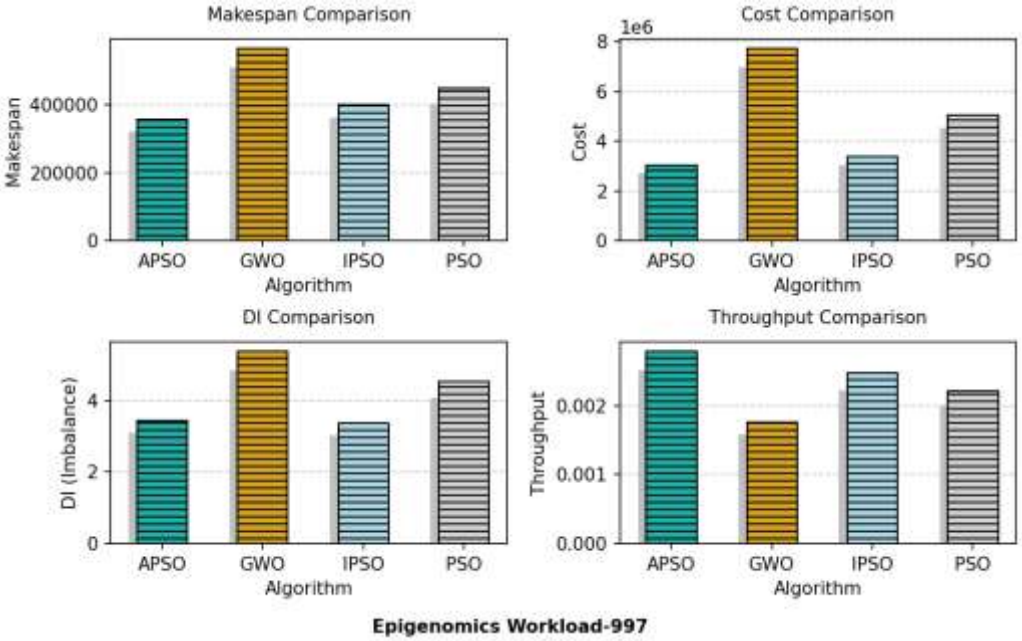

Figure 4. Epigenomics workload results

Fig. 4 compares the results obtained by proposed APSO, IPSO, GWO and PSO algorithms for Epigenomics workload with 997 tasks. The Epigenomics workflow is most computation and data-intensive workflow compared to others. It takes the larger makespan and cost compared to other workflows. With this type of complex data, the proposed algorithm achieves lower makespan, cost and degree of imbalance compared to other algorithms while achieving higher throughput. This proves that our proposed adaptive mechanism based APSO algorithm is capable to search fine solutions in all regions.

The results suggested that the proposed APSO algorithm distribute the load efficiently to available virtual machines. The results confirm the stability and reliability of our proposed APSO algorithm. The inertia weight namely SMIW along with adaptive cognitive and social coefficients c1 and c2 balances the exploration and exploitation effectively. This indicates that the proposed APSO learnt better scheduling pattern compared to other scheduling algorithm namely GWO, PSO and IPSO.

Further the results are also given in Table 3 regarding best, worst and mean values obtained for all 20 times runs. Based on the average results, the proposed APSO reduced 14% makespan in case of CyberShake workflow, 14% in case of Montage workflow, 10% in case of Inspiral workflow, and 11% in case of Epigenomics over its competitor IPSO. In case of cost, the proposed APSO reduced 49% cost in case of CyberShake workflow, 53% in case of Montage workflow, 29% in case of Inspiral workflow, and 10% in case of Epigenomics over its competitor IPSO. In case of degree of imbalance (DI), the proposed APSO reduced 27% DI in case of CyberShake workflow, 31% in case of Montage workflow, 12% in case of Inspiral workflow, and comparable in case of Epigenomics over its competitor IPSO. Also, the proposed APSO improved higher throughput in all cases.

Table 3. All Results Summary of Makespan, Cost and DI

Workload	Metric	Result Type	APSO	GWO	IPSO	PSO
CyberShake	Makespan	Mean	**1093.42**	2297.22	1279.60	1492.12
		Best	**993.21**	2009.21	1035.16	1293.36
		Worst	**1221.96**	2544.88	1579.15	1695.71
	Cost	Mean	**5009.58**	33142.69	9871.53	15087.01
		Best	**3539.01**	25831.49	6790.18	10733.43
		Worst	**6696.51**	39408.59	14230.59	18281.07
	DI	Mean	**3.34**	5.81	4.59	5.10
		Best	**2.61**	5.48	3.40	3.97
		Worst	**4.10**	6.32	5.88	6.03
Montage	Makespan	Mean	**606.15**	1230.43	705.00	763.80
		Best	**520.77**	1138.46	578.16	633.59
		Worst	**685.08**	1365.90	874.04	866.59
	Cost	Mean	**3191.01**	19568.47	6825.93	8434.38
		Best	**2101.66**	17286.45	3632.78	5848.37
		Worst	**4046.76**	22173.34	14344.95	11866.99
	DI	Mean	**3.38**	6.36	4.94	5.42
		Best	**2.96**	5.98	3.75	4.09
		Worst	**3.90**	6.73	6.97	6.51
Inspiral	Makespan	Mean	**17106.16**	28206.36	19071.64	21481.12
		Best	**15001.75**	25275.55	17214.24	19612.95
		Worst	**19034.05**	30262.51	21391.63	23811.57
	Cost	Mean	**127420.21**	417239.47	180381.03	245060.82
		Best	**79003.89**	347827.43	142488.39	199539.91
		Worst	**151792.43**	486211.71	257957.91	299081.63
	DI	Mean	**3.44**	5.91	3.92	4.57
		Best	**2.59**	5.39	3.17	3.64
		Worst	**4.74**	6.43	6.07	5.64
Epigenomics	Makespan	Mean	**356846.10**	567291.64	402245.77	449667.25
		Best	**290799.61**	493539.26	343181.20	371610.71
		Worst	**403658.14**	730762.55	462084.60	518542.99
	Cost	Mean	**3034809.88**	7731040.67	3384390.94	5045721.24
		Best	**2242304.28**	5892528.90	2762891.45	4298473.81

		Worst	**3544773.50**	13541182.50	3849098.84	6218645.60
	DI	Mean	3.45	5.39	**3.37**	4.53
		Best	**2.85**	4.75	2.91	3.64
		Worst	4.46	7.77	**4.06**	6.02

5.2 Convergence Analysis

This section presents the convergence analysis of all algorithm used in this research. Fig. 5 (a-d) presents the convergence behaviour of proposed APSO, IPSO, PSO and GWO algorithm for all workloads CyberShake, Montage, Epigenomics and Inspiral. The x-axis represents the number generations or iterations while the y-axis represents best fitness value obtained during optimization. The convergence curve demonstrate that the proposed APSO algorithm represented by the Black line achieves faster convergence and achieves the lowest fitness values compared to GWO, PSO and IPSO algorithm. The SMIW inertia weight along with adaptive c1 and c2 parameters of APSO make it effective for exploration in early generations and enhanced exploitations in later stages. The introduced adaptive methods prevent the APSO from premature convergence unlike GWO and PSO. Therefore, the convergence analysis proves that proposed APSO is most stable and reliable algorithm among all algorithms in case of all workloads.

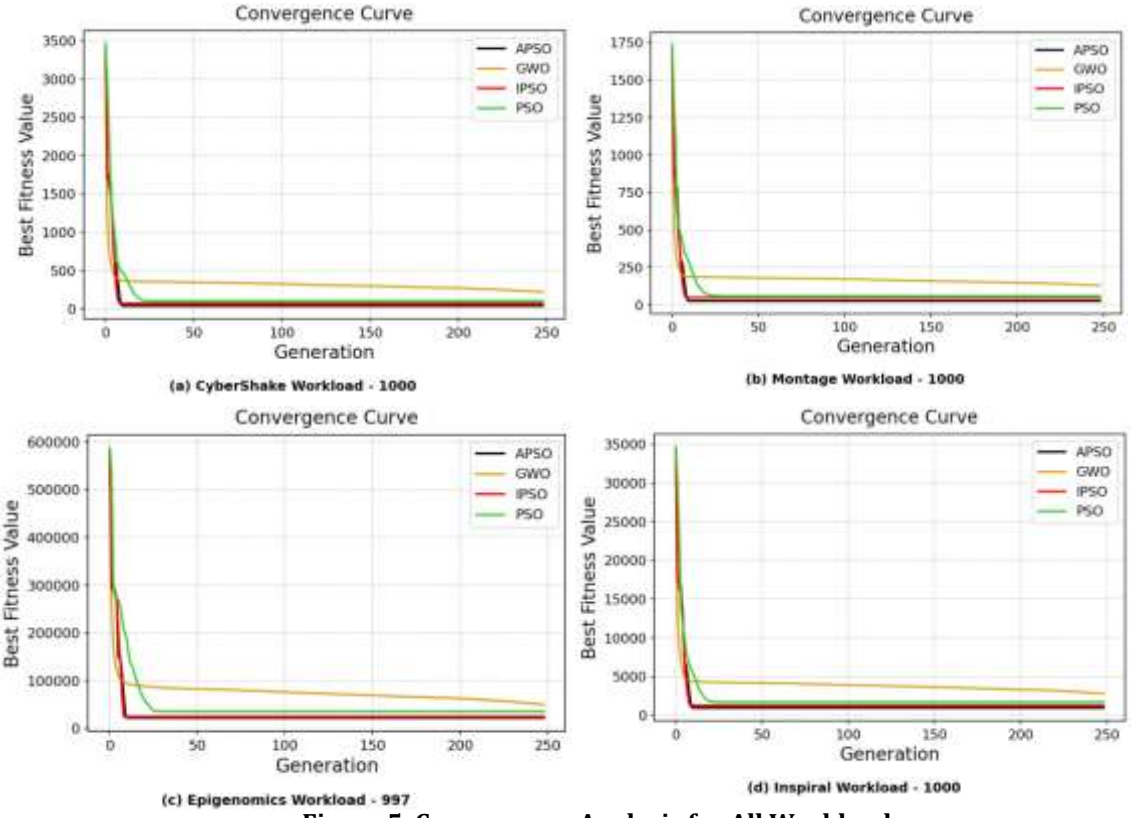

Figure 5. Convergence Analysis for All Workloads

6. Conclusion and Future Scope

Resource management that means efficient resource allocation in cloud computing users' incoming tasks is one of the major issues. Efficient resource management improves the overall system performance and put the impact on computing cost, degree of imbalance etc. Therefore in this paper, we have proposed an efficient resource allocation technique namely APSO. The proposed APSO algorithm is a composed of SMIW inertia weight i.e. exponential decay factor and adaptive c1 and c2 coefficients. This adaptive mechanism helps the APSO algorithm to balance the exploration and exploitation phases due to which the algorithm is capable to find out the best solutions compared to its competitor IPSO, PSO and GWO algorithms. To prove the capability of the proposed APSO algorithm, the experiments have been conducted 20 times. Based on these experiments, the proposed APSO reduced 14% makespan in case of CyberShake workflow, 14% in case of Montage workflow, 10% in case of Inspiral workflow, and 11% in case of Epigenomics over its competitor IPSO. In case of cost, the proposed APSO reduced 49% cost in case of CyberShake workflow, 53% in case of Montage workflow, 29% in case of

Inspiral workflow, and 10% in case of Epigenomics over its competitor IPSO. In case of degree of imbalance (DI), the proposed APSO reduced 27% DI in case of CyberShake workflow, 31% in case of Montage workflow, 12% in case of Inspiral workflow, and comparable in case of Epigenomics over its competitor IPSO. Also, the proposed APSO improved higher throughput in all cases.

In future, the proposed algorithm can be improved by integrating any efficient technique or a new algorithm can be designed. The proposed algorithm can also be tested on independent tasks like Google Traces or independent synthetic tasks. Also, the proposed algorithm cab also be applied to any other area or objectives of cloud computing.

References

[1] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, vol. 53, no. 4, pp. 50–58, 2010.
[2] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality," Future Generation Computer Systems, vol. 25, no. 6, pp. 599–616, 2009.
[3] M. A. Tawfeek, A. El-Sisi, A. Keshk, and F. Torkey, "Cloud Task Scheduling Based on Ant Colony Optimization," International Arab Journal of Information Technology, vol. 12, no. 2, pp. 129–137, 2015.
[4] G. Li, G. Wu, D. Li, and S. Arunagiri, "An Improved Ant Colony Optimization Task Scheduling Algorithm for Grid Computing," Future Internet, vol. 11, no. 4, pp. 90–102, 2019.
[5] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, "A Particle Swarm Optimization-Based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments," 24th IEEE AINA, 2010.
[6] A. Verma and S. Kaushal, "Deadline-Constrained Workflow Scheduling Using Particle Swarm Optimization," Journal of Supercomputing, vol. 68, no. 2, pp. 1–20, 2014.
[7] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
[8] R. Kumar, A. Singh, and P. Sharma, "Task Scheduling Algorithm Using Grey Wolf Optimization Technique in Cloud Computing Environment," 2025.
[9] X. Liu et al., "Cloud Computing Task Scheduling Based on Improved PSO-ACO Algorithm," ACM International Conference Proceedings, 2024.
[10] X. Liu et al., "Cloud Computing Task Scheduling Strategy Based on Improved ACO-PSO Algorithm," 2024.
[11] P. Singh, R. Sharma, and K. Verma, "Hybrid GWO-PSO Based Task Scheduling for Cloud Computing," International Journal of Intelligent Systems, 2020.
[12] K. Rajakumari and S. Mala, "Fuzzy-Based Ant Colony Optimization Scheduling in Cloud Computing," Computer Systems Science and Engineering, vol. 40, no. 2, pp. 671–687, 2021.
[13] C. Chandrashekar et al., "Hybrid Weighted Ant Colony Optimization for Cloud Task Scheduling," Applied Sciences, vol. 13, no. 6, 2023.
[14] H. Prasad et al., "Reinforcement Learning Assisted Grey Wolf Optimization for Cloud Task Scheduling," 2025.
[15] R. Buyya, J. Broberg, and A. Goscinski, Cloud Computing: Principles and Paradigms, Wiley, 2011.
[16] I. Gupta, D. Kumar, and P. K. Jana, "Energy-Aware Task Scheduling Using Particle Swarm Optimization in Cloud Computing," Journal of Grid Computing, vol. 18, no. 4, pp. 785–804, 2020.
[17] A. Kaur and S. Kinger, "Modified Particle Swarm Optimization Based Resource Allocation in Cloud Computing," International Journal of Computer Applications, vol. 95, no. 1, pp. 15–21, 2014.
[18] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent Advancements in Resource Allocation Techniques for Cloud Computing Environment: A Systematic Review," Cluster Computing, vol. 20, no. 3, pp. 2489–2533, 2017.
[19] P. Kumar and M. Venkatesan, "Hybrid Grey Wolf and Ant Colony Optimization for Efficient Cloud Task Scheduling," International Journal of Intelligent Engineering and Systems, vol. 14, no. 5, pp. 357–369, 2021.
[20] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients," IEEE Transactions on Evolutionary Computations, vol. 8, no. 3, pp. 240-255, 2004.
[21] 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.
[22] J. Bhagwan, S. Rani, Manoj, S. Godara, Yashasvi, and S. Kumar, "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, (in Press), 2026.
[23] 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, 2021.

[24] 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.
[25] 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, 1-14, 2014.
[26] Y. Zhao, "Enhanced Butterfly Optimization Algorithm for Task Scheduling in Cloud Computing Environments," International Journal of Advanced Computer Science and Applications, 15(12), 435–443, 2024.