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

SACSO: An Intelligent Algorithm for Energy Efficient Workload Scheduling in Cloud Computing

Jai Bhagwan[1], Manoj[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.
First Author: drjaicse@gmail.com, ORCID: 0000-0002-8708-4029
[2]State Institute of Engineering & Technology, Panchkula, Haryana, India.
*Corresponding Author: manojkoslia91@gmail.com

Abstract

Cloud computing provides scalable and on-demand computational resources, but the increasing energy consumption of cloud datacenters has made energy-efficient task scheduling an important research challenge. The task scheduling problem becomes more complex in heterogeneous cloud environments because minimizing execution time alone may increase the number of active resources and consequently increase energy consumption. This study proposes a Smooth Adaptive Cat Swarm Optimization (SACSO) algorithm for energy-efficient task scheduling in cloud computing. The proposed approach integrates a Smooth Adaptive Inertia Weight (SAIW) mechanism into the tracing mode of Cat Swarm Optimization (CSO) to achieve an effective balance between exploration and exploitation and to reduce the possibility of premature convergence. The scheduling model jointly considers makespan and energy consumption as the primary optimization objectives. The proposed SACSO is implemented in Java using the CloudSim 3.0 simulator and evaluated against CSO, LDCSO, Grey Wolf Optimization (GWO), and Particle Swarm Optimization (PSO). Experiments are conducted using three scientific workflow workloads, namely CyberShake, Inspiral, and Sipht, with 1000 tasks in each workload and heterogeneous virtual machines. Each algorithm is executed for 15 independent rounds to assess the stability of the obtained results. The experimental results demonstrate that SACSO consistently achieves lower makespan and energy consumption than the compared algorithms across all three workloads. Compared with LDCSO, SACSO reduces makespan by 11.66%–22.69% and energy consumption by 15.11%–20.69%. SACSO also achieves the highest throughput among the evaluated algorithms. Convergence analysis further demonstrates that SACSO reaches high-quality solutions rapidly during the initial iterations and maintains stable convergence during subsequent iterations. The results indicate that the proposed SAIW-based tracing mechanism effectively improves the exploration-exploitation balance of CSO and provides an efficient approach for energy-aware task scheduling in heterogeneous cloud environments.

Keywords: Cloud Computing, Cat Swarm Optimization, CSO, Energy, Makespan, SACSO, Virtual Machines (VMs).

1 Introduction

Cloud computing has become an essential computing paradigm for delivering scalable and on-demand computational resources through shared datacenter infrastructures. Virtualization enables cloud providers to dynamically allocate computing resources to a large number of users and applications. This model improves resource utilization and reduces infrastructure costs. Still, cloud datacenters have resulted in extensive electricity consumption and environmental impact. Consequently, energy efficient has arisen as a most important objective in cloud resource management and task scheduling [1] [2]. Task scheduling determines the mapping of user tasks to available virtual machines (VMs) or physical resources. The problem of task scheduling becomes complex when a large number of task-resource combinations is evaluated, especially when workload variations and heterogeneous VMs are considered. The conventional scheduling algorithms generally focus on minimizing makespan and cost. However, a scheduling algorithm with minimum execution time may require more active resources and consequently consumes more energy. Therefore, energy-efficient task scheduling requires a balance between energy consumption and performance related objectives such as makespan, response time, cost, resource utilization etc [3][4]. A few techniques have been developed for energy consumption in cloud environments. Dynamic Voltage and Frequency Scaling (DVFS) adjusts processor frequency according to workload requirements and can reduce power consumption without completely surrendering performance [5][6]. VM consolidation is another important strategy in which workloads are concentrated on fewer physical machines so that underutilized servers can be

switched to low-power or inactive states [2]. But, aggressive consolidation can increase resource conflicts, migration overhead, execution time and SLA violations. Hence, scheduling decisions need to consider the trade-off between energy savings and application performance. The energy-efficient scheduling problem becomes more challenging when heterogeneous resources and multiple objectives are considered. Ding et al. [6] demonstrated that processor heterogeneity and DVFS significantly impact energy-efficient VM scheduling. Dong et al. [7] formulated energy-efficient task assignment as an optimization problem with the objective of minimizing the number of active servers while satisfying time constraints. Workflow scheduling introduces additional complexity because task precedence relationships must also be preserved [8][9]. Thus, a simple heuristic method may become inadequate for obtaining high-quality solutions in large and dynamic cloud environments. Therefore, Swarm Intelligence (SI) and population-based metaheuristic algorithms have gained significant attention for cloud scheduling.

Particle Swarm Optimization (PSO) [10] uses collective behaviour of particles to search for promising solutions. Ant Colony Optimization (ACO) [11] uses pheromone-based cooperative search and is particularly suitable for discrete nature problems. Grey Wolf Optimization (GWO) [12] and Whale Optimization Algorithm (WOA) [13] provide alternative population-based mechanisms for balancing exploration and exploitation. These algorithms can search large scheduling spaces more effectively than other conventional heuristics techniques. Premature convergence and computational overhead remain important challenges. Recent research has increasingly moved toward multi-objective and hybrid swarm-intelligence algorithms. For example, Liu et al. [14] developed an energy-efficient ant colony system for VM placement, while Zhai et al. [15] combined immunity mechanisms with ACO for energy-efficient cloud task scheduling. Rani and Garg [16] proposed an adaptive PSO approach for simultaneously reducing makespan and energy consumption. More recently, Khan and Rasool [17] developed a multi-objective GWO-based task scheduler that jointly optimizes schedule length, energy consumption, and monetary cost. Sanaj et al. [18] proposed EcoTaskOpt a hybrid ACO-PSO framework for heterogeneous cloud scheduling. It is demonstrating the recent trend toward combining complementary swarm mechanisms.

Despite these developments, existing algorithms still face difficulties in simultaneously achieving low energy consumption, makespan minimization, low cost, and high resource utilization. Individual swarm algorithms may become trapped in local optima while simple hybridization does not necessarily provide an effective exploration-exploitation balance. Furthermore, many existing approaches are evaluated under relatively limited workload and resource configurations. These observations motivate the development of improved swarm-intelligence-based task scheduling techniques capable of producing energy-efficient schedules while maintaining competitive execution performance and energy optimization.

The main contributions of this research paper are summarized as:

1. A scheduling model is developed to jointly consider makespan and energy utilization for efficient workflows scheduling.
2. A Smooth Adaptive Inertia Weight (SAIW) is adopted [28] as a nonlinear version of Liner Decreasing Inertia Weight strategy for smoother exploration and exploitation transition during optimization.
3. The SAIW is integrated into tracing mode of Cat Swarm Optimization with the aim to make balance between seeking and tracing modes i.e. exploration and exploitation.
4. The newly designed SACSO algorithm is compared with CSO, GWO and PSO algorithms using makespan and energy consumption as performance measures.

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

2 Literature Review

Research on energy-efficient task scheduling has progressed from conventional resource-management techniques toward intelligent and swarm-based optimization.

Buyya et al. [1] established architecture for energy-efficient cloud management and emphasized dynamic resource provision for reducing datacenter energy consumption in addition to satisfying QoS requirements. Beloglazov et al. [2] developed a heuristic approach for energy-aware resource-allocation based on VMs consolidation, migration and overhead detection. Their work proved that the VM consolidation is an important mechanism for reducing the number of active physical machines which lowers the energy consumption. Calheiros et al. [3] introduced CloudSim, a simulation framework for modelling and simulation cloud infrastructure. This simulator provides VM allocation and scheduling policies. CloudSim widely adopted for evaluation of energy-aware scheduling algorithms under controlled experimental conditions. Topcuoglu et al. [4] proposed the HEFT algorithm for scheduling the tasks on heterogeneous computing resources. Its ranking mechanism has become important baseline for subsequent energy-aware workflow scheduling. Wu et al. [5] proposed a green scheduling algorithm by integration of DVFS with cloud

jobs scheduling. Their approach adjusted processor frequency and selected the resources according to workload requirements. The results demonstrated that DVFS can reduce energy consumption without sacrificing execution performance. Ding et al. [6] introduced energy-efficient VM scheduling under deadline constraints in heterogeneous cloud environments. Their work used the performance-power features of physical machines and incorporated DVFS and resource consolidation for energy minimization under deadlines constraints. Dong et al. [7] formulated an energy-efficient task scheduling problem using a greedy scheduling approach that reduces the number of active servers with service-time constraints. Pietri et al. [8] investigated energy-constrained provision for scientific ensembles and considered energy with budget and deadline constraints. The experiments showed that energy-aware provisioning can satisfy application constraints without disturbing workflow completion. Juarez-Perez et al. [9] designed a dynamic energy-aware scheduling approach for parallel task-based applications. Their proposed approach estimated energy consumption and dynamic resource allocation.

Kennedy and Eberhart [10] introduced PSO which became one of the most widely used population-based optimizer for scheduling purposes. PSO algorithm represents candidate schedules as particles and iteratively improves them using individual and collective search behaviour. Dorigo et al. [11] introduced ACO algorithm which provide solutions using artificial Ants behaviour. ACO is famous for task scheduling because task-resource mapping is fundamentally a discrete optimization problem. Mirjalili et al. [12] proposed GWO algorithm which is a population-based optimizer inspired by the hunting behaviour of grey wolves. Its simple exploration and exploitation phases have made it attractive for cloud-based task scheduling. Mirjalili and Lewis [3] designed WOA algorithm with is another nature-inspired optimizer based on the bubble-net hunting behaviour of humpback whales. Both algorithms have widely adapted for scheduling and resource allocation problems in cloud. Liu et al. [14] proposed an energy-aware ant colony system for VM placement in cloud computing environment. Their approach integrated order-exchange and migration-based local search with the aim to reduce the number of active servers. The experiments on homogeneous and heterogeneous VM-placement problems showed significant energy reduction compared to other conventional heuristic and evolutionary methods. Zhai et al. [15] introduced an immunity-based ant colony mechanism for energy-efficient task scheduling. The proposed method combined immune system approach with ACO to optimization various objectives. Energy efficiency and execution time were main targets. The study demonstrates the potential of hybridizing swarm intelligence with complementary optimization methods. Rani and Gard [16] designed an energy-efficient adaptive method named EE-PSO for independent tasks scheduling. The method used adaptive inertia weight and learning coefficients together with mutation. The research reported improvements over max-min, min-min and GA in terms of makespan and energy reduction. Ding et al. [19] designed a Q-learning-based task scheduling method that combines adaptive VM selection method and task ordering. The aim was the study was to reduce response time and improve server utilization while reducing energy consumption. This work demonstrates the transition from static metaheuristic optimization to adaptive intelligent scheduling. Khorsand and Ramezanpour [20] developed an energy-efficient task allocation method based on multi-criteria decision making approach. The work highlights the significance of multiple scheduling criteria instead of energy optimization independently. Taghinezhad-Niar et al. [21] investigated energy-aware workflow scheduling under budget and deadline constraints. The algorithm was considered for both DVFS-enabled and non-DVFS resources and showed that practical workflow scheduling requires the balance of energy with user-defined constraints. Kumar et al. [22] proposed a PSO-based workflow scheduling method for makespan minimization and energy consumption minimization while maintaining workflow precedence rules. The research's results further demonstrated that the PSO is successful technique for energy-aware scheduling technique. Khan and Rasool [17] designed a multi-objective GWO task scheduling algorithm which minimized schedule length, energy consumption and monetary cost. The approach adapted weighted sorting method for suitable solutions and found better than other scheduling methods. Sanaj et al. [18] recently proposed EcoTaskOpt algorithm which is a hybrid algorithm for task scheduling. The ACO method helps in exploration whereas PSO contributed for velocity-based search. The hybrid formula explicitly targeted the exploration and exploitation trade-off and used energy, makespan and resource utilization as main metrics. Finally, Mehta and Gupta [23] designed a hybrid GWO and improved PSO algorithm with Pareto-knee selection method. The researchers targeted energy-efficient load balancing in cloud environment. Their work reflects the current research trends of combining different swarm-based methods with multi-objective selection strategies.

Overall, the reviewed researches proved that swarm intelligence can effectively address the large and complex search space of cloud's task scheduling. However, conventional PSO, ACO, GWO and other swarm intelligence algorithms can suffer from premature convergence and parameters sensitivity. Recent hybrid methods improved these limitations but there is a scope of improvement using adaptive exploration and exploitation strategies and efficient local search. In particular, simultaneous energy consumption and system performance (makespan) and other objectives remain a challenging research problem. This gap provides a strong motivation for developing an improved swarm-intelligence-based energy-efficient task scheduling algorithm. Recent work has been conducted using Cat Swarm Optimization algorithm [27]. Cat Swarm Optimization algorithm is also suffering from premature convergence problem due to lack of balance between seeking and tracing modes, which needs to be improved further [26].

3 System Model and Problem Definition

This section is discussing problem formulation and system modelling.

3.1 Problem Formulation

Let $W = \{T_1, T_2, T_3, T_4, T_5, \dots, T_n\}$ be the workflow or group 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. The scheduler must regulate a mapping $f: F \rightarrow VM$ such that each task is assigned to one VM. The example can be seen with 15 tasks and 3 VMs in Table 1. In case of workflows, the parent and child execution rules should be maintained. We have considered a data-center having one host and 70 heterogeneous virtual machines for workload scheduling in cloud computing system.

Table 1: Task-VM Mapping 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 system, the total completion time taken by all tasks in a workflow or task set is called as makespan [25][26]. Its mathematical model is represented in Eq. (1).

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

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

3.3 Energy Model

The energy model [1][24] used in this research is a VM-level dynamic-idle power consumption model that accounts for both the idle time period and the active time period processing of each VM, with an additional MIPS-dependent workload factor. The idle time for a VM v is the difference between overall makespan and its actual processing time, the calculation is shown in Eq. (2).

$$T_v^{idle} = T - T_v^{proc} \tag{2}$$

Where, T_v^{idle} represents idle time of a VM and T_v^{proc} represents active time of a VM.

To make the realistic cloud environment, the MIPS-dependent energy factor is involved because if number of VMs is increased or decreased, the energy gets affected. This factor is given in Eq. (3).

$$F_v = 1 + \alpha \left(\frac{MIPS_v}{MIPS_{base}}\right) \tag{3}$$

Therefore, the total energy consumed by VM v is given in Eq. (4).

$$E_v = \frac{P_{idle} T_v^{idle} + P_{max} T_v^{proc} F_v}{3.6 \times 10^6} \tag{4}$$

Where, P_{max} is the maximum power used by a VM i.e. 100%, P_{idle} is idle power consumption which is considered 70% in this research. The denominator converts energy Jules into kWh.

The total energy consumption of the cloud environment is given in Eq. (5).

$$E_{total} = \sum_{v=1}^{m} \frac{P_{idle} T_v^{idle} + P_{max} T_v^{proc} F_v}{3.6 \times 10^6} \tag{5}$$

3.4 Throughput Model

In this research, we have used throughput metric [24] to detect the system's performance and it is computed using Eq. (6).

$$\text{throughput} = \text{Total Tasks/Makespan} \tag{6}$$

3.5 Fitness Model

We have combined makespan and energy metrics in a single objective fitness function (F_x) which is calculated using Eq. (7). We have used number of datacenters and VMs to normalize the fitness values. In this research, the values of makespan and energy are minimized.

$$F_x = \frac{w_1 \times \text{makespan} + w_2 \times \text{energy}}{\text{DC} \times \text{VMs}} \tag{7}$$

Where, DC is datacenter, VMs are virtual machines, w_1 and w_2 are the weights used to give the priorities to makespan and energy consumption. Here, more priority is given to makespan to stable the system performance as end users need performance. The values of w_1 and w_2 are 0.6 and 0.4 correspondingly given that $w_1 + w_2 = 1$.

4 Proposed Method

This section describes the method we have designed in this research.

4.1 CSO Theory

The Cat Swarm Optimization is a swarm intelligence algorithm developed by Chu et al. in 2006 [26]. For hunting purposes, it has two modes seeking and tracing. Both the modes are discussed below:

1) Seeking Mode – This mode uses four parameters namely SMP (seeking memory pool), SRD (seeking range dimension), CDC (count dimension to change) and SPC (self-position considerations). A MR (mixing ratio) value is used to distribute the cats in seeking and tracing modes. The SM (seeking mode) can be understood by following steps:
   - Make SMP copies of the Q^{th} cat.
   - Update the position using SRD value given in Eq. (8).

$$X_{Q,D} = (1 + rand \times SRD) \times X_{Q,D} \tag{8}$$

     Where, $X_{Q,D}$ is cat's position and rand is random variable between (0, 1).
   - Evaluate fitness and find best cat.
   - Swap the original cat with best copy.
2) Tracing Mode – In this mode, the cat moves rapidly towards its food by updating velocity and position in order to exploit the target food. The TM (tracing mode) can be understood by following steps:
   - Calculate velocity and position of Q^{th} cat using Eq. (9).

$$V_{Q,D} = \omega \times V_{Q,D} + \left(c1 \times r1 \times \left(X_{BEST,\ D} - X_{Q,D}\right)\right) \tag{9}$$

     Where, $V_{Q,D}$ is velocity of the cat, c1 is learning coefficient, r1 is a random number between (0, 1), and ω is inertia weight which may be set 0.5 i.e. static in basic algorithm.
   - The position of the cat Q^{th} is updated using Eq. (10).

$$X_{Q,D} + X_{Q,D} + V_{Q,D} \tag{10}$$

     Where, $X_{Q,D}$ is position of the Cat_Q in dimension D.

4.2 SAIW Theory

We have adopted Smooth Adaptive Inertia Weight (SAIW) [28] which is a modified version of LDIW (Liner Decreasing Inertia Weight) strategy. This makes an effective balance between seeking and tracing modes compared to LDIW, because it searches for more exploration than LDIW inertia in early stages. The decay behaviour of both inertia strategies can be seen in Fig. 1.

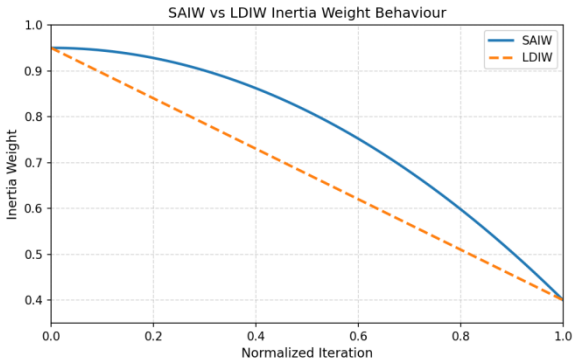

Figure 1: Inertia Weight Strategies Behaviour

The impact of these strategies can be seen in the results and discussion section as well. The mathematical model of the SAIW is given in Eq. (11).

$$\omega(g) = \omega_{max} - (\omega_{max} - \omega_{min}) \left(\frac{g}{g_{max}}\right)^{\gamma} \tag{11}$$

Where, $\omega(t)$ is inertia weight at iteration or generation g, ω_{max} represents maximum inertia weight (starting point) and ω_{min} represents the minimum inertia weight, γ is a nonlinear decay parameter which is set to 2.0 in this research. This inertia weight called SAIW has been integrated in tracing mode of CSO algorithm to make it more efficient, the algorithm is discussed in coming section.

4.3 HEFT Theory

The HEFT theory [26] is famous one to maintain the DAG dependency and estimate the earliest finish time for workload scheduling. The HEFT algorithm can calculate the earliest finish time for both types of dependent and independent tasks to set task priorities as shown in Algorithm 1.

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 SACSO Algorithm

In the proposed SACSO algorithm, the SAIW inertia weight has been injected in the TM (tracing mode) of the Cat Swarm Optimization for better balancing between seeking and tracing modes i.e. exploration and exploitation.

Algorithm 2: SACSO Algorithm

Input: Solution cat, HEFT Generated Populations i.e. Cats, SMP, CDC, MR, max_Generations etc.
Output: Best Schedule gBest

```
For Gen = 0 to max_Generations Do
    Update Seeking and Tracing Modes using MR value
    For each cat in Cats Do
        If cat is in Seeking Mode Then
            run Seeking Mode    // Update positions, use Eq. (8)
        Else
            // Use Eq. (11) for inertia weight in TM
            run Tracing Mode    // Update positions, use Eq. (10)
        End If
        If cat.fitness < gBest.fitness Then
            gBest ← copy (cat)
        End If
    End For
End For
return Best Schedule (gBest)
```

The working of the proposed algorithm can be understood by the following steps:

1) First of all the population is initialized using HEFT theory, and all required parameters are initialized.
2) Then the SM and TM modes are updated based on MR value which is set to 0.2 i.e. 80% cats are distributed in seeking mode and 20% in tracing mode.
3) The seeking mode is responsible to find out new solutions and tracing mode exploits the target i.e. the algorithm moves towards best solution quickly using tracing mode.
4) After this, the fitness is checked and best solution is stored in the memory.
5) This process continues till the condition satisfied i.e. maximum generations are reached i.e. 200.

5 Simulation Results and Discussion

The simulation experiments are conducted using CloudSim 3.0 simulator written in Java programming language. 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 3 types of 70 heterogeneous VMs having computing power between 500-1000 MIPS, 512-1024 MB RAM and 1000 Mbps bandwidth. All the algorithms have been run for 15 rounds to check the stability. The rest simulation parameters are described in Table 2. The performance parameters are makespan, energy and throughput as described earlier.

Table 2: Simulation Parameters of All Algorithms

Proposed SACSO and LDCSO Algorithms Properties	
Parameter	**Value**
Max Generations	200
No. of Cats (Population)	50
SMP, SRD, CDC, MR	5, 0.3, 0.3, and 0.2 respectively
c1, rand, ω_{max}, ω_{max}, and γ (γ in case of SACSO only)	1.5, (0, 1), 0.95, 0.4, and 2.0 respectively
CSO Algorithm Properties	
Parameter	**Value**
Max Generations	200
No. of Cats	50
SMP, MR, SRD, CDC, c1, ω	5, 0.2, 0.3, 0.3, 1.5 and 0.5 Respectively
PSO and GWO Algorithms Properties	
Parameter	**Value**
Generations	200
No. of Particles	50
c_1, c_2, ω (in case of PSO only)	1.5, 1.5, 0.5

The workloads [25][26] used in this research are CyberShake, Inspiral, and Sipht each containing 1000 tasks. Table 3 is representing best, worst and average makespan and energy utilized by all algorithms including proposed SACSO. For clear understanding and better visualization the results are also discussed with graphs in Figs. 2, 3 and 4.

Table 3: Summary of Makespan and Energy Utilization

Workload	**Metric**	**Result Type**	**SACSO**	**LDCSO**	**PSO**	**GWO**	**CSO**
CyberShake-1000	Makespan	Mean	1280.979	1656.916	9980.618	6452.840	3560.701
		Best	1202.667	1233.994	9466.072	5650.655	1270.007
		Worst	1385.986	1949.745	10433.291	9339.462	5430.560
	Energy	Mean	1.963	2.475	13.785	9.006	5.067
		Best	1.857	1.900	13.085	7.914	1.947
		Worst	2.106	2.874	14.401	12.932	7.612
Inspiral-1000	Makespan	Mean	21583.603	24433.176	102692.100	68321.305	44354.257
		Best	20427.639	19873.233	100179.291	63949.858	20423.223
		Worst	23727.574	28716.727	105741.675	73000.596	58161.943
	Energy	Mean	31.583	35.457	141.777	95.236	62.575
		Best	30.023	29.247	138.360	89.274	30.003
		Worst	34.521	41.278	145.936	101.619	81.345
Sipht-1000	Makespan	Mean	25054.970	29916.692	84123.722	57547.521	55460.571
		Best	22767.354	22540.294	81612.068	52565.699	21586.633
		Worst	27550.508	34141.099	86808.284	64992.802	66108.687
	Energy	Mean	35.777	42.394	116.024	80.021	77.149
		Best	32.673	32.328	112.603	73.232	31.048
		Worst	39.155	48.154	119.674	90.165	91.633

Figure 2 shows the average makespan of proposed SACSO, LDCSO CSO, GWO and PSO algorithms across three workloads. From the results, it can be observed that the makespan is reduced by proposed SACSO algorithm for all three workloads. PSO has achieved the highest makespan among all algorithms due to premature convergence. The makespan is improved from 11.66% to 22.69% by SACSO compared to LDCSO algorithm. The SACSO algorithm achieves better performance due to effective balance between seeking and tracing modes and effective convergence

as shown in Fig. 5. In initial stages, it explores the optimized VMs mapping to workloads i.e. scheduling solutions and later it moves to better solution quickly.

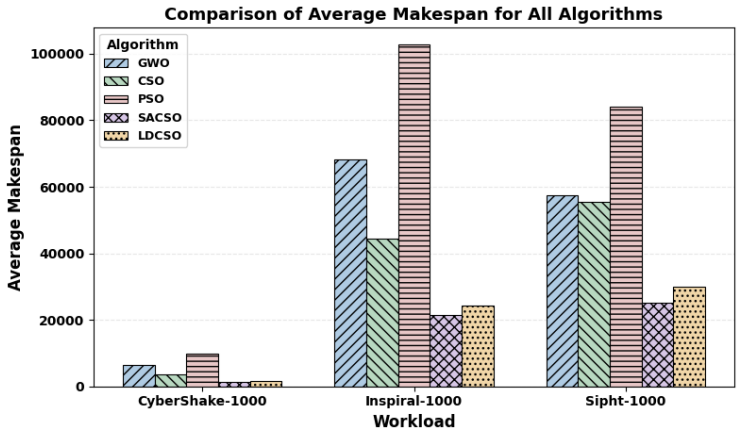

Figure 2: Makespan Comparison

Figure 3 compares the average energy consumption of the five algorithms across three scientific workloads. The proposed SACSO algorithm is indicating better energy efficiency as it has the lowest energy consumtion for all three workloads. PSO consumes the higest energy compared to others due to its poor convergence. The results proved that the proposed SACSO improves the energy consumption from 15.11% to 20.69% compared to second most effective algorithm i.e. LDCSO. The improvement percentage is also displyed in Table 4 for clear understanding. This is because the SAIW (Smooth Adaptive Inertia Weight) makes the balance between exploration and exploitation. In the initial iterations, the proposed SACSO algorithm explores the solutions in effective manner and in later stages it moves faster toward best solution.

Figure 4 compares the throuphut achieved by proposed SACSO, LDCSO, GWO, CSO and PSO algorithms across the three workloads. The results reveal that the most productive algorithm is SACSO, which produces highest throughputs among all.

Table 4: Percentage of Improvement of SACSO over LDCSO

Workload	Makespan Improvement (%)	Energy Improvement (%)
CyberShake-1000	22.69	20.69
Inspiral-1000	11.66	10.93
Sipht-1000	16.25	15.61

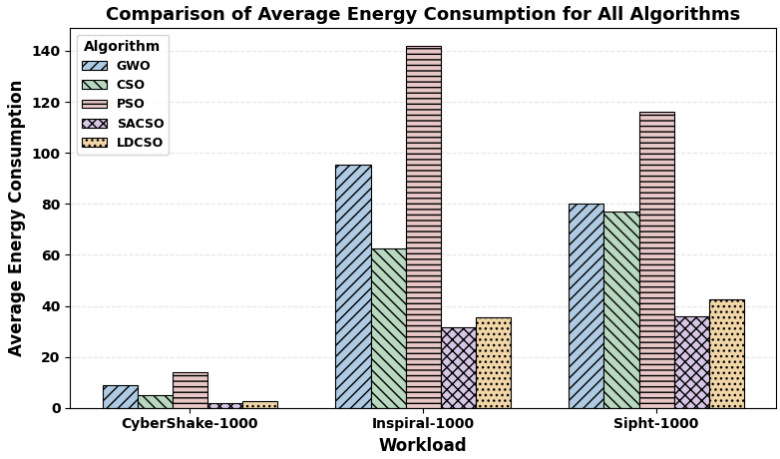

Figure 3: Energy Consumption Comparison

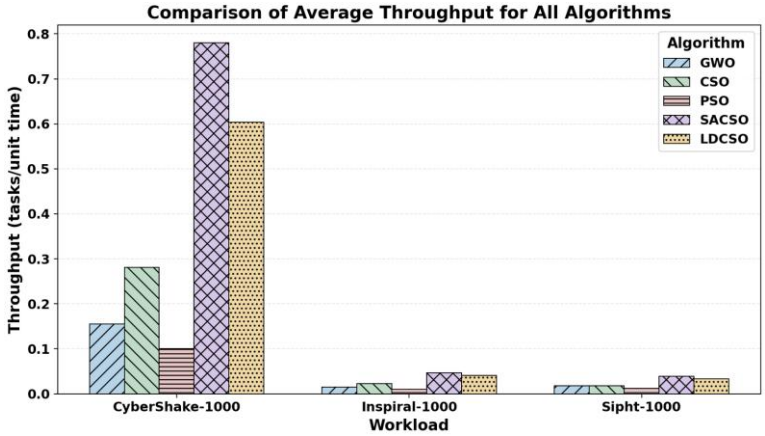

Figure 4: Throughput Comparison

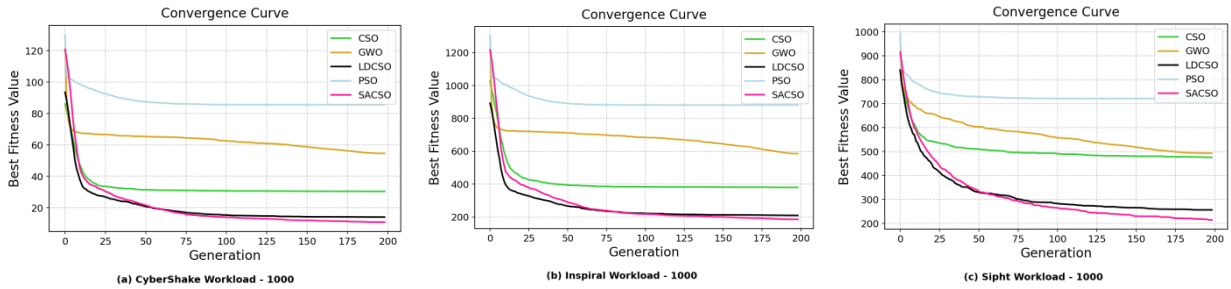

Figure 5: Convergence Results of All Algorithms

Figure 5, having three convergence analysis show the change in best fitness value with increasing generations for SACSO, LDPSO, CSO, GWO and PSO on all three workloads. All algorithms show a sharp reduction in fitness during the first few generations. The SACSO curve (deep pink colour) reaches a lower fitness value faster than other algorithms in all three workloads. After rapid improvement in initial evolution SACSO becomes relatively stable. It indicates that the SACSO algorithm reaches a good solution without substantial late-stage oscillations. This effect is due to the velocity update using SAIW inter weight, which helps the proposed algorithm to search effective solutions.

6 Conclusions and Future Direction

This research presented a Smooth Adaptive Cat Swarm Optimization (SACSO) algorithm for energy-efficient task scheduling in heterogeneous cloud computing environments. The proposed method was developed to address the limitations of conventional swarm-intelligence algorithms, particularly premature convergence and the difficulty of maintaining an effective balance between exploration and exploitation. A scheduling model considering makespan and energy consumption was formulated, and a Smooth Adaptive Inertia Weight (SAIW) mechanism was incorporated into the tracing mode of Cat Swarm Optimization to improve its search behaviour. The proposed SACSO algorithm was implemented using the CloudSim 3.0 simulator and evaluated against LDCSO, conventional CSO, GWO, and PSO using the CyberShake, Inspiral, and Sipht scientific workloads, each containing 1000 tasks. Heterogeneous virtual machines were used to provide a realistic resource environment, and each algorithm was executed for 15 rounds to examine the stability of the results. The experimental results demonstrate that SACSO consistently provides superior scheduling performance across the evaluated workloads. In particular, SACSO reduces makespan by 11.66%–22.69% compared with LDCSO. Similarly, the proposed method achieves a reduction of 15.11%–20.69% in energy consumption compared with the second-best performing LDCSO algorithm. SACSO also achieves the highest throughput among the evaluated algorithms, demonstrating that the improvement in energy efficiency does not come at the expense of overall scheduling productivity. The convergence analysis further supports the effectiveness of the proposed approach. SACSO achieves a rapid reduction in fitness during the early iterations and subsequently becomes relatively stable, indicating that it can identify promising scheduling solutions quickly while avoiding substantial late-stage oscillations. The improved convergence behaviour can be attributed to the SAIW mechanism, which provides a smoother transition between exploration and exploitation within the

tracing mode. The results therefore indicate that integrating adaptive inertia control with CSO can effectively improve its search capability and scheduling performance.

Overall, the experimental findings demonstrate that SACSO is an effective approach for balancing execution performance and energy efficiency in heterogeneous cloud task scheduling. The proposed method outperforms the considered conventional and modified swarm-intelligence algorithms on the evaluated workloads and provides a promising solution for energy-aware cloud resource management. In future work, the proposed algorithm can be extended to multi-objective scheduling by incorporating additional objectives such as monetary cost, response time, resource utilization, SLA violations, and load balancing. Its performance can also be investigated in larger-scale and dynamic cloud environments, real-world Google workload traces, and edge-fog-cloud infrastructures. Furthermore, adaptive parameter learning and hybrid local-search mechanisms may be explored to further improve the robustness and scalability of SACSO.

References

1. R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges," Proceedings of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications, 2010.
2. A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing," Future Generation Computer Systems, vol. 28, no. 5, pp. 755–768, 2012.
3. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23–50, 2011.
4. H. Topcuoglu, S. 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.
5. C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters," Future Generation Computer Systems, vol. 37, pp. 141–147, 2014.
6. Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," Future Generation Computer Systems, vol. 50, pp. 62–74, 2015.
7. Z. Dong, N. Liu, and R. Rojas-Cessa, "Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers," Journal of Cloud Computing, vol. 4, article 5, 2015.
8. I. Pietri, M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, and R. Sakellariou, "Energy-constrained provisioning for scientific workflow ensembles," Proceedings of the 2013 IEEE 3rd International Conference on Cloud and Green Computing, pp. 34–41, 2013.
9. F. Juarez, J. Ejarque, and R. M. Badia, "Dynamic energy-aware scheduling for parallel task-based application in cloud computing," Future Generation Computer Systems, vol. 78, pp. 257–271, 2018.
10. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95—International Conference on Neural Networks, 1995.
11. M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics, vol. 26, no. 1, pp. 29–41, 1996.
12. S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014.
13. S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51–67, 2016.
14. X.-F. Liu, Z.-H. Zhan, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "An energy efficient ant colony system for virtual machine placement in cloud computing," IEEE Transactions on Evolutionary Computation, vol. 22, no. 1, pp. 113–128, 2018.
15. J. Zhai, X. Liu, and H. Zhang, "Energy-efficient cloud task scheduling research based on immunity-ant colony algorithm," Lecture Notes in Computer Science, vol. 11063, pp. 490–501, 2018.
16. R. Rani and R. Garg, "Energy efficient task scheduling using adaptive PSO for cloud computing," International Journal of Reasoning-based Intelligent Systems, vol. 13, no. 2, pp. 50–58, 2021.
17. M. A. Khan and R. ur Rasool, "A multi-objective grey-wolf optimization based approach for scheduling on cloud platforms," Journal of Parallel and Distributed Computing, vol. 187, article 104847, 2024.
18. M. S. Sanaj, M.-F. Horng, S. S. Shankar, C.-C. Lo, R. Sunder, U. K. Lilhore, et al., "Energy-efficient task scheduling in cloud computing with EcoTaskOpt: A hybrid ant colony and particle swarm optimization approach," International Journal of Computational Intelligence Systems, vol. 19, article 64, 2026.
19. D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning based dynamic task scheduling for energy-efficient cloud computing," Future Generation Computer Systems, vol. 108, pp. 361–371, 2020.

20. R. Khorsand and M. Ramezanpour, "An energy-efficient task-scheduling algorithm based on a multi-criteria decision-making method in cloud computing," International Journal of Communication Systems, vol. 33, no. 9, e4379, 2020.
21. A. Taghinezhad-Niar, S. Pashazadeh, and J. Taheri, "Energy-efficient workflow scheduling with budget-deadline constraints for cloud," Computing, vol. 104, no. 3, pp. 601–625, 2022.
22. A. Kumar, S. Ghosh, B. B. Naik, and P. Kuila, "Energy efficient workflow scheduling in cloud computing systems using particle swarm optimization," Proceedings of the International Conference on Signal Processing and Computer Vision (SIPCOV 2023), pp. 266–278, 2024.
23. V. Mehta and A. Gupta, "A hybrid grey wolf and improved particle swarm optimization technique with Pareto knee selection for energy efficient load balancing in cloud computing," Discover Applied Sciences, vol. 8, article 337, 2026.
24. M. M. Nezami and A. Kumar, "Energy efficient workflow allocation in cloud computing using improved grey wolf optimization," International Journal of Advanced Computer Science and Applications, vol. 16, no. 10, pp. 491-498, 2025.
25. 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.
26. J. Bhagwan, S. Rani, S. Kumar and S. Godara, "DWCSO: a hybrid intelligent algorithm for multi-type workload scheduling in cloud computing," International Journal of Computer Information Systems and Industrial Management Applications, vol. 18, no. 10s, pp. 227-244, 2026.
27. A. Gunduz, S. Senan and Z. Gurkas-Aydin, "A hybrid DECSO algorithm for efficient multi-objective task scheduling," Cluster Computing, vol. 28, article 848, 2025.