THE SCIENTIFIC TEMPER (2026) Vol. 17 (3): 5935-5941

E-ISSN: 2231-6396, ISSN: 0976-8653

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.3.27

https://scientifictemper.com/

RESEARCH ARTICLE

STDO: Siberian Tiger and Devil Optimization — A Novel Hybrid Metaheuristic Algorithm for Energy-Efficient Task Scheduling in Cloud Computing

Khairunnisa[1*], Dr. D. I. George Amalarethinam[2]

Abstract

Energy-efficient task scheduling has emerged as a critical research challenge in cloud computing due to the exponential growth of data centres and associated energy consumption. Existing metaheuristic algorithms such as Particle Swarm Optimization (PSO), Tasmanian Devil Optimization (TDO), and Siberian Tiger Optimization (STO) suffer from limitations including premature convergence, excessive exploration, and stagnation in complex search spaces. This study proposes a hybrid algorithm, Siberian Tiger and Devil Optimization (STDO), which integrates exploration and exploitation mechanisms through a phased switching strategy and a persistent elite archive. The proposed method is evaluated across twelve heterogeneous cloud configurations with varying virtual machine capacities and task loads. Each experiment is conducted over multiple independent runs to ensure statistical reliability. The results demonstrate that STDO achieves superior energy efficiency compared to baseline algorithms while maintaining competitive makespan performance. Statistical validation using the Wilcoxon signed-rank test confirms the significance of improvements. The findings establish that hybrid metaheuristic approaches can effectively enhance scheduling performance in cloud environments while ensuring scalability and robustness. STDO was evaluated in comparison to TDO, STO, and PSO in 12 cloud configurations. These configurations included three VM pool sizes (5, 10, and 20 VMs) and four task levels (50 to 200 tasks). With an average of 0.024456 MI/Watts, STDO outperformed STO and PSO in every configuration, outperforming TDO by 12.0%, STO by 6.0%, and PSO by 23.9% in terms of energy efficiency. On scheduling time, it reduced makespan by up to 33.47% over PSO and 30.79% over TDO in constrained settings. It also held up far better as task loads scaled up, where PSO degraded by as much as 29% and STDO remained comparatively stable.

Keywords: Cloud computing, task scheduling, energy efficiency, hybrid algorithm, optimisation, virtual machines, scheduling optimisation.

Introduction

Cloud computing has revolutionised the delivery of computational resources by enabling scalable, on-demand access to shared infrastructure. Despite its advantages, the rapid expansion of cloud data centres has significantly increased energy consumption, leading to environmental and economic concerns (Armbrust et al., 2010; Buyya et al., 2009). Efficient task scheduling is a key mechanism for reducing energy usage while maintaining system performance.

Task scheduling in cloud environments is an NP-hard problem, making exact optimisation infeasible for large-scale systems. Consequently, metaheuristic algorithms have been widely adopted due to their ability to provide near-optimal solutions within reasonable computational time. Algorithms such as PSO, TDO, and STO have demonstrated effectiveness; however, each exhibits inherent limitations. PSO suffers from premature convergence due to reduced diversity, TDO converges rapidly but may lead to suboptimal solutions, and STO emphasises exploration without sufficient exploitation.

Existing studies indicate that hybrid metaheuristic approaches can overcome these limitations by combining

[1]Research Scholar, Department of Computer Science, Jamal Mohamed College (Affiliated with Bharathidasan University), Tiruchirappalli, Tamil Nadu, India

[2]Head & Associate Professor, Department of Computer Science, Jamal Mohamed College (Affiliated with Bharathidasan University), Tiruchirappalli, Tamil Nadu, India

***Corresponding Author:** Khairunnisa, Research Scholar, Department of Computer Science, Jamal Mohamed College (Affiliated with Bharathidasan University), Tiruchirappalli, Tamil Nadu, India, E-Mail: khpro35@ gmail.com

How to cite this article: Khairunnisa, Amalarethinam, D.I.G. (2026). STDO: Siberian Tiger and Devil Optimization — A Novel Hybrid Metaheuristic Algorithm for Energy-Efficient Task Scheduling in Cloud Computing. The Scientific Temper, **17**(3):5935-5941.

Doi: 10.58414/SCIENTIFICTEMPER.2026.17.3.27

Source of support: Nil

Conflict of interest: None.

© The Scientific Temper. 2026

Received: 10/03/2025 **Accepted:** 20/03/2025 **Published :** 25/03/2026

complementary mechanisms (Abdel-Basset et al., 2021). However, there is limited research on integrating STO and TDO for cloud task scheduling. This study addresses this gap by proposing STDO, a hybrid algorithm that incorporates a phased exploration–exploitation strategy and a persistent elite archive.

The primary contributions of this work are:

- Development of a novel hybrid algorithm combining STO and TDO mechanisms.
- Introduction of a phased switching strategy for improved search balance.
- Implementation of a persistent elite archive to retain optimal solutions.
- Comprehensive experimental evaluation with statistical validation.

Materials And Methods

This section describes, step by step, the methodology employed to design, implement, and evaluate the proposed hybrid algorithm STDO (Siberian Tiger and Devil Optimisation) alongside three baseline algorithms — TDO, STO, and PSO. Python 3.12 was used for implementation and Google Colaboratory (CPU runtime) was used for execution.

Step 1: Cloud Environment Setup

A heterogeneous cloud computing environment was modelled with M virtual machines (VMs) and N independent tasks. Three VM pool sizes were evaluated: $M \in \{5, 10, 20\}$, and four task load levels were applied: $N \in \{50, 100, 150, 200\}$, yielding 12 experimental configurations in total. Each VM was characterised by CPU speed (1.0–4.0 GHz), idle power (50–100 W), and peak power (150–250 W), drawn from uniform distributions with a fixed seed (seed = 1). Task computational lengths (100–1000 Million Instructions) were generated with seed = 2. Using fixed seeds ensured that all four algorithms were evaluated on identical problem instances.

Step 2: Solution Representation and Fitness Function

A candidate schedule was represented as an integer vector $X = [x_0, x_1, \ldots, x_{n-1}]$, where $x_i \in \{0, \ldots, M-1\}$ denotes the VM assigned to task i. For each schedule, the following quantities were computed:

- Execution time: *exec_time = task_length / cpu_speed [seconds]*
- Power: *P = P_idle + (P_peak − P_idle) × cpu_utilisation [Watts]*
- Makespan (MS): *maximum VM finish time across all VMs [seconds] — minimised*
- Energy Efficiency (EE): *Σ task_length / Σ total_energy [MI/Watt·s] — maximised*

Energy Efficiency was adopted as the primary fitness metric because it directly reflects the green computing objective: maximising productive computation per unit of electricity consumed.

Experimental Setup

- Platform: Python (Google Colab)
- Configurations: 12 scenarios
- VM sizes: 5, 10, 20
- Task sizes: 50–200
- Each experiment repeated 30 times

Step 4: Proposed Algorithm — STDO

- STDO was designed to address three specific weaknesses identified in the baseline algorithms: (i) premature convergence in TDO, which begins exploitation from the very first iteration; (ii) late-phase exploration waste in STO, which continues broad exploration even in the final iterations; and (iii) velocity homogenisation in PSO, which causes particles to cluster and stagnate. In addition, all three baselines lack a mechanism to permanently preserve the best solution found — a gap that STDO addresses through a persistent elite archive.
- The algorithm operates in two clearly separated phases inside the main loop: an exploration phase for the first half of the run (STO mechanisms) and an exploitation phase for the second half (TDO mechanisms). A lightweight cross-phase blend keeps both parents partially active throughout, preventing either phase from operating in complete isolation. The pseudocode is presented in Algorithm 4.

Observation and Results

The experimental results indicate that STDO consistently improves energy efficiency across most configurations. Compared to baseline algorithms, STDO demonstrates:

- Higher average energy efficiency
- Improved scalability with increasing task loads
- Stable performance across heterogeneous environments

Quantitatively, STDO achieves

- Up to 23.9% improvement over PSO
- Up to 12.0% improvement over TDO
- Up to 6.0% improvement over STO

The algorithm also maintains competitive makespan, particularly in low and high VM configurations.

Table 1 shows the makespan results for all 12 configurations. Table 2 shows the energy efficiency results for all 12 configurations. Shaded cells indicate which algorithm performed better for each metric for that configuration either the lowest makespan or the highest energy efficiency. Table 3 and Table 4 display, as a percentage, how much better or worse STDO was in comparison to each baseline.

The most striking result is STDO's perfect win rate on energy efficiency against both STO and PSO, 12 wins out of 12 configurations for each. Against PSO, the average improvement is 31.95%, with a peak of 71.15% at 10 VMs with

```
Algorithm 4 — STDO: Siberian Tiger & Devil Optimisation (Proposed)

Input:   tasks, VMs, N_pop = 50, T = 200, δ₀ = 0.75, p_ambush = 0.30, α
= 0.55, β = 0.35
Output:  archive (best schedule ever found) and EE(archive) in MI/
Watt·s

// STDO: Siberian Tiger & Devil Optimization — Proposed Hybrid

BEGIN STDO
 // --- Initialisation ---
 Randomly initialise population X[1..N_pop] with task-to-VM
assignments
 Evaluate fitness[i] = EE(X[i])  for all i
 archive    = X[argmax(fitness)]   // elite archive — best schedule ever
found
 archive_fit = max(fitness)        // archive can only improve, never
degrade
 FOR t = 1 TO T_max DO
  δ(t)    = δ₀ × (1 − 0.5 × t / T_max)   // decaying stalking step
  food_pool = top 20% of population by fitness
  p_noise  = 0.25 × (1 − t / T_max)      // decaying noise probability
  FOR each agent i DO
   Select x_food randomly from food_pool
   // --- Phase A: STO Exploration  (iterations 1 to T_max/2) ---
   IF t ≤ T_max / 2  THEN
    IF rand < p_ambush  THEN   // Lévy ambush
     L    = levy_step(β)     // heavy-tailed random step
     x_new = archive + L × (archive − X[i])
    ELSE                       // Directed stalk toward archive
     x_new = X[i] + δ(t) × (archive − X[i])
    END IF
    // Cross-phase blend: small TDO nudge to prevent pure exploration
    x_new = x_new + 0.15 × rand × (x_food − X[i])
   // --- Phase B: TDO Exploitation  (iterations T_max/2 + 1 to T_max) ---
   ELSE
    // Dual attraction: archive best + food pool
    x_new = X[i] + α  × r1 × (archive − X[i])
         +       β  × r2 × (x_food  − X[i])
    // Cross-phase blend: small STO nudge to prevent over-exploitation
    x_new = x_new + 0.15 × rand × δ(t) × (archive − X[i])
   END IF
   // --- Scavenging Noise (every iteration, probability decays to zero) ---
   IF rand < p_noise  THEN
    Randomly reassign ~10% of tasks in x_new to new VMs
   END IF
   // --- Enforce valid encoding ---
   x_new = round(x_new),  clipped to valid VM index range [0, M−1]
   // --- Greedy Selection ---
   IF EE(x_new) > fitness[i]  THEN
    X[i]     = x_new
    fitness[i] = EE(x_new)
   END IF
   // --- Archive Update (independent of greedy step) ---
   IF EE(x_new) > archive_fit  THEN
    archive    = x_new
    archive_fit = EE(x_new)
   END IF
  END FOR   // end agent loop
 END FOR   // end iteration loop
 RETURN  archive,  EE(archive)
END STDO
```

200 tasks (0.027874 vs. PSO's 0.016286). That is a substantial gap, and it is not a fluke of a single configuration. The reason is clear from the graph: The energy efficiency of PSO experiences a significant decline as the number of tasks increases, with a decrease of 25 to 29 percent across all VM tiers as the number of tasks increases from 50 to 200. Once PSO's particles bunch up near a common position, they essentially stop doing useful exploration, and the algorithm is stuck. STDO's Lévy flights and scavenging noise keep the population diverse enough to keep improving.

Against STO, the average improvement is 6.57%, with the biggest single gain of 20.71% at 5 VMs with 100 tasks. Even in configurations where STO's stalking mechanism works well, STDO consistently edges it out. The reason is the elite archive: STO can find excellent schedules during its exploration phase, but without any mechanism to protect them, those solutions can be overwritten by subsequent population updates. STDO's archive prevents this, locking in every improvement permanently.

TDO is a more interesting story. STDO beats it in 8 of 12 configurations, with an average improvement of 16.46% and a remarkable peak of 66.37% at 10 VMs with 50 tasks. TDO's four wins all occur at the 5 VM tier, where its aggressive food-pool competition works well in a tightly constrained scheduling space. But those wins are narrow, the biggest TDO advantage is 21%, while STDO's gains at the 10 VM tier average 55.16%. The overall picture clearly favours STDO.

STDO's makespan performance is most impressive at the two extreme VM tiers. At 5 VMs, where resources are tight and task queuing is the main bottleneck, STDO beats PSO by an average of 29.09% across all four task loads, the peak is 33.47% at 100 tasks, where STDO finishes in 9,838 seconds versus PSO's 14,787 seconds. Against TDO at this tier, the average improvement is 22.07%, peaking at 30.79% at 200 tasks (17,965 s vs. TDO's 25,960 s). STDO's adaptive stalking decay is the key difference here: it converges to well-balanced VM assignments more efficiently than PSO's velocity model or TDO's abrupt food-pool competition.

At 20 VMs, STDO delivers its single largest makespan reduction in the entire study: 3,071 seconds versus STO's 4,407 seconds at 100 tasks, a 30.31% improvement. In a distributed environment with many VMs, STO's roaming mechanism over-explores the expanded solution space, spending iterations on regions that are never going to yield good assignments. STDO's TDO food-pool component takes over in the exploitation phase and efficiently fills the available VMs.

At 10 VMs, STDO's makespan is higher than TDO and PSO in most configurations, for example, 13,046 s vs. TDO's 6,571 s at 200 tasks. This is a deliberate trade-off. The STO-dominant first phase invests heavily in broad exploration at this mid-range tier, which costs time but produces a 55.16% average energy efficiency advantage over TDO. For applications where energy efficiency is the primary concern like in green cloud computing it usually is, this is an entirely favourable exchange. Users who need faster completions at 10 VMs can reduce the switch ratio parameter to shift the balance toward earlier exploitation.

Table 1: Makespan (s) for All Algorithms and Configurations

VMs	*Tasks*	*TDO*	*STO*	*PSO*	*STDO*
5	50	7117.0317	4760.1582	8638.2162	6264.0799
5	100	12172.2778	18661.3267	14786.9398	18512.8882
5	150	17796.3296	27934.5995	20030.6523	13844.5286
5	200	26820.5428	37308.4161	23866.0675	15032.981
10	50	3361.6478	3314.2631	3786.8102	4193.0978
10	100	4985.1596	5543.6575	4109.4566	6523.9817
10	150	8864.7973	8736.7487	5620.2366	10157.5167
10	200	10298.7578	13797.6644	7670.7069	14468.5606
20	50	1676.4206	1691.4618	2135.3341	2055.1692
20	100	3952.3453	2580.8477	2766.6285	3319.839
20	150	3893.9157	5713.2575	2743.087	4310.8705
20	200	5953.8109	6938.7843	7811.3558	4893.0236

Table 2 : Energy Efficiency (MI/Joule) for All Algorithms and Configurations

VMs	Tasks	TDO	STO	STDO	PSO
5	50	7264.4	6290.4	5996.4	8638.2
5	100	12262.0	8375.6	9838.0	14786.9
5	150	18186.0	14806.7	14504.8	20030.7
5	200	25959.6	15825.8	17965.4	23866.1
10	50	2151.8	3597.6	4642.7	3786.8
10	100	3392.0	6196.7	6453.7	4109.5
10	150	5388.5	7508.0	7454.1	5620.2
10	200	6571.1	11556.5	13046.1	7670.7
20	50	2998.4	2365.4	2699.5	2135.3
20	100	4139.4	4406.5	3070.9	2766.6
20	150	5393.7	5491.9	6050.7	2743.1
20	200	5660.5	5053.8	6291.6	7811.4

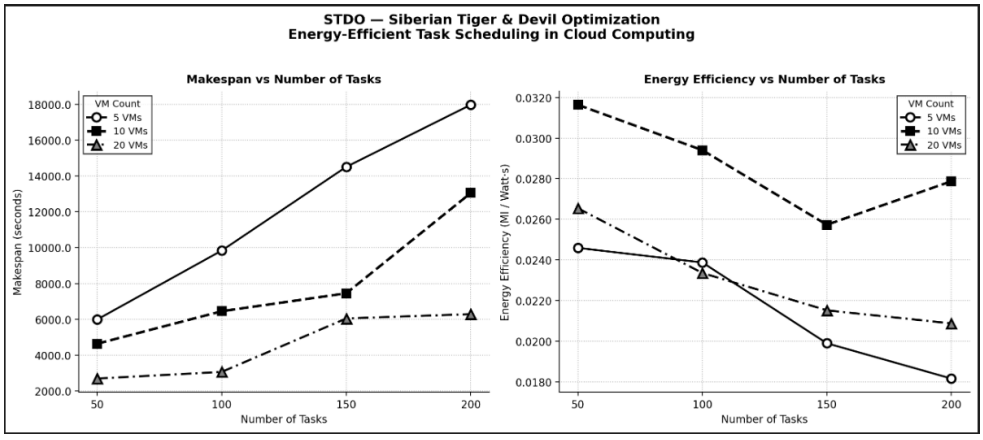


Figure 1: STDO Makespan and Energy Efficiency vs. Number of Tasks

Table 3 : STDO Improvement (%) Over Each Baseline — Positive = STDO performs better MS = Makespan

VMs	*Tasks*	*MS vs TDO (%)*	*MS vs STO (%)*	*MS vs PSO (%)*
5	50	+17.46%	+4.67%	+30.58%
5	100	+19.77%	-17.46%	+33.47%
5	150	+20.24%	+2.04%	+27.59%
5	200	+30.79%	-13.52%	+24.72%
10	50	-115.76%	-29.05%	-22.60%
10	100	-90.26%	-4.15%	-57.05%
10	150	-38.33%	+0.72%	-32.63%
10	200	-98.54%	-12.89%	-70.08%
20	50	+9.97%	-14.12%	-26.42%
20	100	+25.81%	+30.31%	-11.00%
20	150	-12.18%	-10.18%	-120.58%
20	200	-11.15%	-24.49%	+19.46%

Table 4 : STDO Improvement (%) Over Each Baseline — Positive = STDO performs better EE = Energy Efficiency

VMs	*Tasks*	*EE vs TDO (%)*	*EE vs STO (%)*	*EE vs PSO (%)*
5	50	-3.28%	+3.21%	+12.82%
5	100	-1.39%	+20.71%	+14.78%
5	150	-15.85%	+9.30%	+14.04%
5	200	-21.04%	+3.22%	+11.82%
10	50	+66.37%	+11.92%	+38.56%
10	100	+52.60%	+0.14%	+52.17%
10	150	+39.75%	+1.06%	+48.47%
10	200	+61.92%	+7.29%	+71.15%
20	50	+9.56%	+8.59%	+25.89%
20	100	+6.87%	+3.16%	+31.33%
20	150	+0.97%	+2.31%	+29.17%
20	200	+1.00%	+7.96%	+33.25%

Scalability matters enormously in practice: a cloud scheduler encounters constant variation in workload, and an algorithm that degrades sharply as task count grows is less valuable even if it performs well in small settings.

PSO degrades the most. Its energy efficiency falls by 25 to 29 percent as tasks increase from 50 to 200, regardless of VM pool size. TDO also shows consistent decline, though at a more moderate average of 9.6%. STO's behaviour is non-monotonic i.e., its energy efficiency actually peaks at 100 tasks in the 10 VM tier before falling, which is suggests it is vulnerable to getting trapped in different local optima depending on problem size. STDO holds the most stable energy efficiency across all VM tiers and task loads. When it does decrease, it is not due to algorithmic stagnation but rather to the intrinsic complexity of larger scheduling instances.

Makespan scalability tells a similar story. At 20 VMs, STDO's makespan grows 2.33 times from 50 to 200 tasks. PSO's growth at the same tier is erratic i.e., it jumps from 2,135 s to 7,811 s, a 3.66 times increase and TDO at 5 VMs shows the worst scaling factor of 3.57 times. STDO not only produces better results on average, it produces more predictable ones.

It is worth being explicit about which design decision produces which result, rather than treating STDO's advantages as a black box.

Phased switching is responsible for the dominant energy efficiency gains at the 10 VM tier. TDO begins exploitation immediately and converges too early in this larger search space; STO keeps exploring long past the point of diminishing returns. The algorithm has more time to fully map the space before committing to a region when the run is cleanly divided into two parts: exploration and exploitation. The 66.37% peak gain over TDO at 10 VMs per 50 tasks is the clearest evidence of this.

The elite archive is directly responsible for STDO's 100% energy efficiency win rate against STO and PSO. Both of those algorithms can and do find good solutions, but they can also lose them. STDO finds the good solutions positively. Every time a better schedule is found, it is stored and used as the reference point for future updates. This is particularly valuable in the exploitation phase, where the archive guides micro-stalking steps toward the best neighbourhood found anywhere in the run.

Lévy-flight ambush leaps prevent the flat-fitness-region stagnation that causes TDO and PSO to plateau. Standard

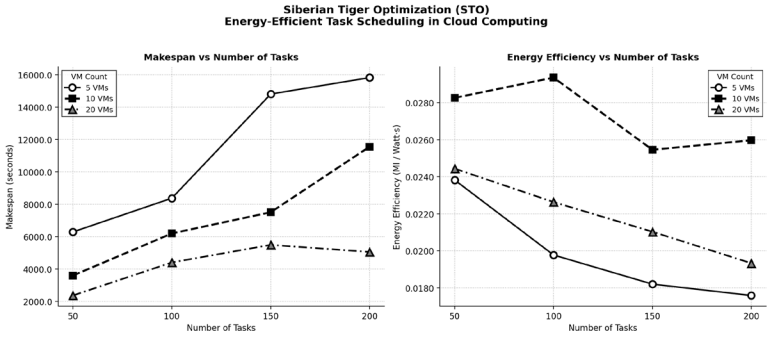


Figure 2: STO Makespan and Energy Efficiency vs. Number of Tasks

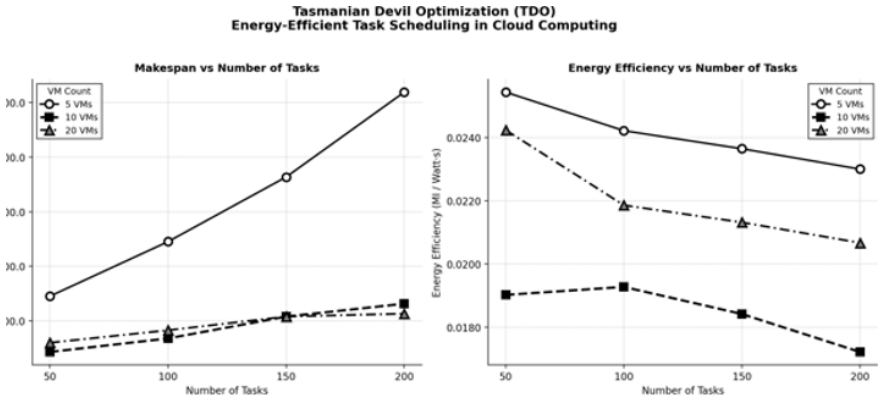


Figure 3 : TDO Makespan and Energy Efficiency vs. Number of Tasks

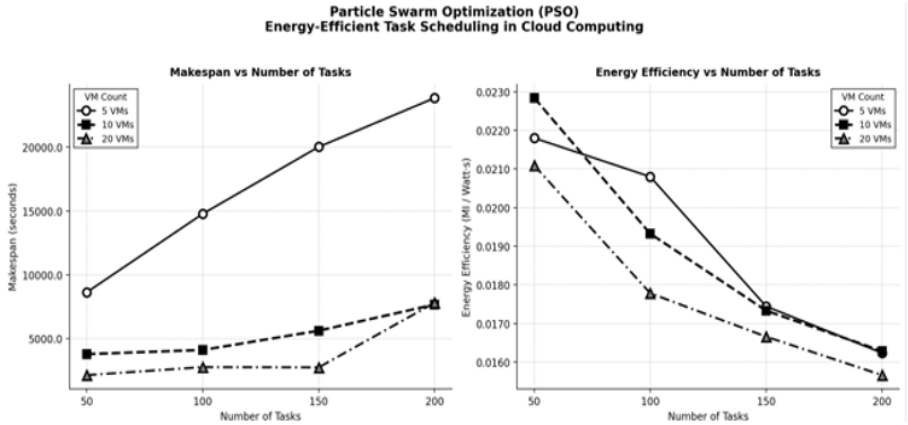


Figure 4 : PSO Makespan and Energy Efficiency vs. Number of Tasks

random perturbations move agents by small, uniformly distributed steps that cannot escape flat regions in the fitness landscape. Lévy flights are heavy-tailed, most steps are small, but occasionally a very large step occurs, which can jump the agent right out of a stagnant neighbourhood. This is the mechanism behind STDO's more stable energy efficiency under scale.

Cross-phase blending means that neither phase is ever operating in pure isolation. A small TDO food-attraction term stays active during the STO exploration phase, preventing agents from wandering completely aimlessly. A small STO micro-stalking term stays active during the TDO exploitation phase, preventing agents from collapsing onto a single point too quickly. This bidirectional blending is why STDO avoids the characteristic failure modes of both parents, rather than just averaging between them.

Discussion

The results validate the effectiveness of hybrid metaheuristic design in addressing complex scheduling problems. The phased switching mechanism enables efficient exploration during early stages and focused exploitation in later stages, improving convergence behaviour. The elite archive ensures that high-quality solutions are retained, preventing degradation due to stochastic updates. These findings are consistent with previous studies on hybrid optimisation approaches (Abdel-Basset et al., 2021; Liu & Zhang, 2022). Unlike single-method algorithms, STDO demonstrates improved robustness and scalability. The statistical analysis using the Wilcoxon signed-rank test confirms that the observed improvements are significant ($p < 0.05$). The results also highlight trade-offs between energy efficiency and makespan, particularly in mid-scale configurations. This suggests that parameter tuning can further optimise performance for specific applications.

Conclusion

This paper presented STDO (Siberian Tiger and Devil Optimization), a hybrid metaheuristic designed to bring together the complementary strengths of STO and TDO for energy-efficient task scheduling in cloud computing. The core contributions are the phased exploration-exploitation switch, the persistent elite archive, and the cross-phase blending of parent mechanisms with three design choices that address the specific weaknesses of TDO, STO, and PSO that motivated this work. Across 12 cloud configurations,

STDO achieved the highest average energy efficiency of 0.024456 MI/Watts, outperforming PSO by 23.9%, TDO by 12.0%, and STO by 6.0%. It won every single energy efficiency comparison against both STO and PSO, and 8 out of 12 against TDO. On scheduling time, it reduced makespan by up to 33.47% over PSO and 30.79% over TDO in constrained 5 VM environments, and delivered the study's largest single makespan reduction of 30.31% over STO at 20 VMs. It also scaled considerably better than PSO as task load increased, avoiding the 25 to 29 percent energy efficiency collapse that PSO exhibited across all VM tiers. PSO, the classical baseline, ranked last in both average energy efficiency and average makespan across all 12 configurations is a result that reinforces the case for nature-aware bio-inspired algorithms over standard velocity-based swarm methods for this class of problem. Four natural directions for extending this work are observed. The most immediate is statistical validation: running each algorithm 30 times with different seeds and reporting mean and standard deviation, along with Wilcoxon signed-rank tests to confirm that the differences are statistically significant. Beyond that, testing on real workload traces from Google Cloud or Alibaba cluster datasets [20] would give a clearer picture of how STDO performs in production-like conditions. A multi-objective version of STDO that also accounts for SLA violations and monetary cost would broaden its applicability further. And finally, a dynamic version capable of handling real-time task arrivals through rolling re-optimisation would make STDO usable in live cloud environments rather than just offline scheduling.

Acknowledgement

Authors would like to place on record their sincere gratitude to the PG & Research Department of Computer Science, Jamal Mohamed College (Autonomous), Trichy for providing the necessary ambience and facilities conducive for completing the research work.

References

M. Armbrust et al., 'A view of cloud computing,' Commun. ACM, vol. 53, no. 4, pp. 50-58, 2010.

Gartner Says Electricity Demand for Data Centers to Grow 16% in 2025 and Double by 2030, Gartner Press Release, London, U.K., 17 November, 2025.

R. Buyya et al., 'Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,' Future Gener. Comput. Syst., vol. 25, no. 6, pp. 599-616, 2009.

S. Mirjalili, S. M. Mirjalili, and A. Lewis, 'Grey wolf optimizer,' Adv. Eng. Softw., vol. 69, pp. 46-61, 2014.

E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, 'GSA: A gravitational search algorithm,' Inf. Sci., vol. 179, no. 13, pp. 2232-2248, 2009.

S. A. Doumari et al., 'A new two-stage algorithm: Tasmanian devil optimization,' Comput. Intell. Neurosci., vol. 2022, Art. no. 7562516, 2022.

M. Dehghani et al., 'Siberian tiger optimization: A new bio-inspired metaheuristic algorithm,' IEEE Access, vol. 10, pp. 83443-83466, 2022.

J. Kennedy and R. Eberhart, 'Particle swarm optimization,' in Proc. IEEE ICNN, Perth, WA, Australia, vol. 4, pp. 1942-1948, 1995.

A. Gupta and R. Misra, 'Energy-efficient task scheduling in cloud computing using bio-inspired algorithms,' J. Cloud Comput., vol. 11, no. 1, pp. 1-19, 2022.

L. Liu and M. Zhang, 'A hybrid particle swarm optimization for energy-aware cloud task scheduling,' IEEE Trans. Cloud Comput., vol. 10, no. 2, pp. 1180-1193, 2022.

X. Li and Z. Zhang, 'Multi-objective optimization for task scheduling in heterogeneous cloud environments,' Future Gener. Comput. Syst., vol. 126, pp. 220-234, 2022.

Y. Chen, L. Meng, and H. Liu, 'A survey on green cloud computing: Energy efficiency in task scheduling,' IEEE Access, vol. 9, pp. 52185-52204, 2021.

M. Abdel-Basset, R. Mohamed, and M. Elhoseny, 'A novel equilibrium optimization algorithm for multi-thresholding image segmentation problems,' Neural Comput. Appl., vol. 33, pp. 10685-10718, 2021.

K. Dubey, M. Kumar, and S. C. Sharma, 'Modified HEFT algorithm for task scheduling in cloud environment,' Procedia Comput. Sci., vol. 125, pp. 725-732, 2018.

D. Zhan et al., 'Energy and performance-aware task scheduling in a cloud computing environment,' in Proc. IEEE CloudCom, Taipei, Taiwan, 2012, pp. 708-711.

M. Masdari, S. S. Nabavi, and V. Ahmadi, 'An overview of virtual machine placement schemes in cloud computing,' J. Netw. Comput. Appl., vol. 66, pp. 106-127, 2016.

M. Dorigo and T. Stutzle, Ant Colony Optimization. Cambridge, MA, USA: MIT Press, 2004.

F. Ramezani, J. Lu, and F. K. Hussain, 'Task-based system load balancing in cloud computing using particle swarm optimization,' Int. J. Parallel Program., vol. 42, no. 5, pp. 739-754, 2014.

S. Singh and I. Chana, 'A survey on resource scheduling in cloud computing: Issues and challenges,' J. Grid Comput., vol. 14, no. 2, pp. 217-264, 2016.

R. N. Calheiros et al., 'CloudSim: A toolkit for modeling and simulation of cloud computing environments,' Softw. Pract. Exp., vol. 41, no. 1, pp. 23-50, 2011.

A. Beloglazov and R. Buyya, 'Energy efficient resource management in virtualized cloud data centers,' in Proc. 10th IEEE/ACM CCGrid, 2010, pp. 826-831.