STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–17.
Published online in International Academic Press (www.IAPress.org)

A Hybrid Sand Cat Swarm Optimization and Particle Swarm Optimization (SCSO–PSO) Algorithm for DDoS-Resilient Task Scheduling in Cloud Computing Environments

Yasser Mohammad Al-Sharo*

Cybersecurity and Cloud Computing, Faculty of Information Technology, Ajloun National University P.O. 43, Ajloun-26810, JORDAN

Abstract Cloud computing has become a cornerstone of modern computational infrastructure, enabling scalable, on-demand resource provisioning for diverse applications. However, the efficiency and reliability of cloud services are increasingly threatened by Distributed Denial-of-Service (DDoS) attacks, which strain computational resources and degrade task scheduling performance. Existing metaheuristic scheduling approaches, while effective under normal conditions, often exhibit significant performance degradation when subjected to adversarial workload injection. This paper proposes a novel hybrid Sand Cat Swarm Optimization–Particle Swarm Optimization (SCSO–PSO) algorithm that integrates the exploration capability of SCSO with the convergence efficiency of PSO through a stagnation-aware switching mechanism. The algorithm is evaluated against Genetic Algorithm (GA), standalone PSO, Artificial Bee Colony (ABC), and the recent GA–PSO hybrid of Kaplan and Babalik under both normal and DDoS-affected cloud environments using CloudSim 3.0.3. Experimental results across six scenarios demonstrate that SCSO–PSO consistently achieves the lowest makespan, with improvements of approximately 1.5–3.0% over GA–PSO under normal conditions and 4.7–6.1% under DDoS conditions. The proposed algorithm also achieves superior CPU resource utilization (67.85–74.86%) and competitive execution times while maintaining statistical significance (Wilcoxon signed-rank test, $\alpha = 0.05$) across all pairwise comparisons. These findings establish SCSO–PSO as a resilient and efficient scheduling framework for adversarial cloud environments.

Keywords cloud computing; task scheduling; Sand Cat Swarm Optimization (SCSO); Particle Swarm Optimization (PSO); hybrid metaheuristic; DDoS attacks; makespan; CloudSim.

DOI: 10.19139/soic-2310-5070-4071

1. Introduction

Cloud computing has firmly established itself as a foundational paradigm of contemporary information technology, providing on-demand access to scalable computational resources, elastic storage, and distributed services across heterogeneous platforms. Its rapid proliferation has transformed how enterprises, governments, and research institutions provision and manage computational workloads, making the efficient scheduling of tasks across virtual machines (VMs) a central concern in cloud resource management [2, 3, 21]. The task scheduling problem in cloud computing is NP-hard, as it requires assigning n independent tasks to m heterogeneous VMs such that one or more objectives—typically makespan minimization—are optimized [17]. The combinatorial search space, which grows as m^n, renders exact methods computationally intractable for practical problem sizes, and has therefore motivated the adoption of metaheuristic optimization algorithms as the dominant solution paradigm [2, 16, 18].

Among the most widely studied metaheuristics for cloud scheduling are Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO), each offering distinct trade-offs between exploration and exploitation [3, 17, 16]. However, a growing body of research

*Correspondence to: Yasser Mohammad Al-Sharo (Email: yaser.shrah@anu.edu.jo). Cybersecurity and Cloud Computing, Faculty of Information Technology, Ajloun National University P.O. 43, Ajloun-26810, JORDAN.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

demonstrates that standalone algorithms are inherently limited: GA suffers from premature convergence due to selection pressure [16], PSO tends to stagnate in local optima during the exploitation phase [17], and ABC, while maintaining population diversity, converges slowly on complex fitness landscapes [2]. These limitations have driven the development of hybrid metaheuristics that seek to combine the complementary strengths of multiple algorithms. Notable examples include the GA–PSO hybrid of Manasrah and Ali [16], the GA–GWO framework of Behera and Sobhanayak [18], and the most recent GA–PSO hybrid by Kaplan and Babalik [1], which additionally evaluated scheduling resilience under DDoS-affected cloud environments.

Despite these advances, a critical gap persists in the literature: none of the existing hybrid metaheuristics for cloud task scheduling have incorporated the Sand Cat Swarm Optimization (SCSO) algorithm [6], a recent swarm-based metaheuristic with strong exploration capability and a low-parameter design that is particularly well suited for dynamic and adversarial environments. SCSO's adaptive sensitivity range mechanism, which transitions agents from exploration to exploitation based on biologically inspired acoustic perception, offers a principled alternative to the fixed exploration–exploitation trade-offs of GA and PSO. Furthermore, the increasing prevalence of DDoS attacks on cloud infrastructures [25, 29, 30] underscores the need for scheduling algorithms that remain effective under adversarial workload injection, a dimension that only Kaplan and Babalik [1] and the Improved Wild Horse Optimizer (A-WHO) [4] have addressed in the context of metaheuristic scheduling.

Moreover, while SCSO offers strong global search capabilities, recent literature confirms that its individual agents tend to stagnate in local optima during the exploitation phase, particularly when applied to high-dimensional or constrained problems [12]. PSO, conversely, is well known for its rapid convergence but suffers from diversity loss in later iterations [17]. The proposed SCSO–PSO hybridization directly addresses these complementary weaknesses: SCSO serves as the primary exploration engine, continuously generating diverse candidate solutions, while PSO provides fast local refinement around promising regions. The stagnation-aware switching mechanism ensures that when PSO's convergence stalls, the best solution discovered by the SCSO swarm is injected into the PSO population, thereby re-introducing diversity precisely when it is most needed. This design philosophy differs fundamentally from fixed-schedule or purely random hybridization schemes, and is particularly suited to the discontinuous fitness landscapes induced by DDoS attacks.

The principal contributions of this study are summarized as follows:

1. A novel hybrid SCSO–PSO algorithm is proposed for cloud task scheduling, representing—to the best of our knowledge—the first application of SCSO within a hybrid metaheuristic framework designed specifically for DDoS-resilient scheduling.
2. A stagnation-aware switching mechanism is introduced to dynamically balance exploration and exploitation, mitigating the local-optima trap inherent to both SCSO and PSO.
3. A systematic comparative evaluation is conducted against GA, PSO, ABC, and the recent GA–PSO hybrid under both normal and DDoS-affected cloud environments, using makespan, resource utilization, and execution time as performance metrics.
4. The study provides empirical evidence that the proposed SCSO–PSO consistently reduces makespan and improves scheduling stability under adversarial conditions, contributing to the broader objective of building resilient cloud infrastructures.

The remainder of this paper is organized as follows. Section 2 reviews related work on metaheuristic task scheduling and DDoS-aware cloud security. Section 3 presents the mathematical formulation of the task scheduling problem and provides background on SCSO and PSO. Section 4 details the proposed SCSO–PSO algorithm, including the stagnation-aware switching mechanism and computational complexity analysis. Section 5 describes the experimental setup, including the CloudSim simulation environment, algorithm parameters, DDoS emulation protocol, and evaluation metrics. Section 6 reports and discusses the experimental results. Section 7 concludes the paper and outlines directions for future work.

2. Related Work

Metaheuristic algorithms have become the dominant approach for addressing the NP-hard nature of cloud task scheduling. Genetic Algorithm (GA), PSO, Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) constitute the most widely studied baselines, each offering distinct trade-offs between exploration and exploitation [2, 3, 17, 21]. GA's crossover and mutation operators provide strong population diversity but impose high computational overhead due to selection, crossover, and mutation operations at each generation [16]. PSO converges rapidly through velocity-based updates guided by personal and global best positions but is susceptible to premature convergence and swarm stagnation in later iterations [17, 3]. ABC maintains a balanced search through employed, onlooker, and scout bee phases but tends to converge slowly on complex, high-dimensional landscapes [2]. These individual limitations have motivated the development of hybrid approaches that combine the complementary strengths of two or more algorithms [14, 15, 23, 5, 20, 27, 19, 26, 28, 31].

A critical comparison of these hybrid approaches reveals important structural differences in their search dynamics. The GA-GWO hybrid of Behera and Sobhanayak [18] relies on the Grey Wolf Optimizer's hierarchical alpha-beta-delta leadership structure, which provides strong exploitation through guided convergence toward leadership positions. However, this hierarchical structure can limit exploration when the alpha solution is trapped in a local optimum, as all lower-ranked wolves are drawn toward it. In contrast, the GA-PSO framework of Kaplan and Babalik [1] alternates between GA's population-level operators and PSO's velocity-based updates, but does so on a fixed schedule without an adaptive mechanism to detect and respond to search stagnation. The proposed SCSO-PSO addresses both limitations: SCSO's non-hierarchical, sensitivity-range-based exploration ensures that agents explore diverse regions independently of the current best, while the stagnation-aware switching mechanism provides an adaptive (rather than fixed) hybridization trigger. An emerging line of complementary work proposes the Improved Wild Horse Optimizer (A-WHO) [4], the GA-PSO hybrid with heuristic initialization [17], the enhanced PSO framework of Pradhan et al. [3], and a Green cloud scheduling approach using hybrid GSCOA [11].

Manasrah and Ali [16] proposed one of the earliest hybrid GA–PSO frameworks for workflow scheduling, demonstrating that the integration of GA's diversification with PSO's rapid convergence reduces makespan and improves load balancing compared with GA, PSO, HSGA, and WSGA. Building on this line of work, Kaplan and Babalik [1] proposed a GA–PSO hybrid specifically evaluated under DDoS-affected cloud conditions using CloudSim, establishing the first systematic benchmark for adversarial scheduling resilience. Their work represents the most directly relevant baseline for the present study and provides the experimental methodology that this paper extends. The quality-of-service-aware coati optimization of Tamilarasu and Singaravel [21] and the multi-strategy sand cat optimization for edge computing [9] further illustrate the growing interest in nature-inspired scheduling for heterogeneous and edge environments.

The Sand Cat Swarm Optimization (SCSO) algorithm, proposed by Seyyedabbasi and Kiani in 2022, is a relatively recent swarm-based metaheuristic inspired by the desert hunting behavior of sand cats, particularly their ability to detect prey through low-frequency sound perception [6]. SCSO is distinguished by its low-parameter design (a single control parameter, the sensitivity range maximum S_M), its adaptive exploration–exploitation transition, and its competitive performance on standard benchmark functions. Since its introduction, SCSO has been extended in several directions: Li and Wang [7] improved SCSO for wireless sensor network coverage optimization; Li et al. [8] adapted SCSO for flexible job shop scheduling; Zhang et al. [22] proposed a multi-strategy improved SCSO for global optimization; Li et al. [12] introduced elite decentralization and crossbar strategies to address SCSO's exploitation-phase stagnation; and a bi-optimized SCSO combined with deep learning was applied to enhanced cloud security [10]. A multi-objective task scheduling approach combining Pelican Optimization and SCSO has also been explored [24]. However, to the best of our knowledge, no prior study has hybridized SCSO with PSO for cloud task scheduling or evaluated its performance under DDoS-affected environments.

While the majority of DDoS-related research focuses on detection and traffic-level mitigation, a more recent line of work investigates the resilience of scheduling mechanisms themselves under adversarial conditions. Kaplan and Babalik [1] represent the most directly relevant contribution, proposing a GA–PSO hybrid that was evaluated under both normal and DDoS-affected environments, with DDoS emulated through resource-demand scaling

in CloudSim. Somani et al. [25] provided a comprehensive taxonomy of DDoS attacks in cloud computing, identifying scheduling as a downstream component affected by resource exhaustion. Bhardwaj et al. [29] surveyed state-of-the-art DDoS mitigation techniques, noting that most solutions operate at the network or traffic level, with little attention paid to the scheduling layer. AlSaleh et al. [30] proposed a Bayesian-CNN-based DDoS detection framework for cloud environments, and Bazm et al. [32] addressed malicious VM detection through clustering-based approaches. These works collectively establish the importance of scheduling-layer resilience as a complement to detection-based defenses.

3. Background and Problem Formulation

This section establishes the theoretical foundation of the proposed work. Section 3.1 formulates the cloud task scheduling problem as a constrained combinatorial optimization problem with makespan minimization as the principal objective. Section 3.2 reviews the Particle Swarm Optimization (PSO) algorithm, and Section 3.3 reviews the Sand Cat Swarm Optimization (SCSO) algorithm, including all equations that form the basis of the proposed hybrid.

3.1. Cloud Task Scheduling Problem

The cloud task scheduling problem considered in this study involves the assignment of a set of independent, non-preemptive tasks (cloudlets) to a set of available virtual machines (VMs) such that the overall completion time, known as the makespan, is minimized. Formally, let $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$ denote the set of n cloudlets, where each cloudlet T_k is characterized by its computational length L_k (measured in millions of instructions, MI). Let $\mathcal{V} = \{V_1, V_2, \ldots, V_m\}$ denote the set of m available VMs, where each VM V_j has a processing capacity P_j (measured in millions of instructions per second, MIPS).

Under these definitions, the execution time of cloudlet T_k when assigned to VM V_j is computed as follows:

$$\mathrm{ET}(T_k, V_j) = \frac{L_k}{P_j} \tag{1}$$

The total execution time on VM V_j is the sum of the execution times of all cloudlets assigned to it:

$$\mathrm{TotalTime}(V_j) = \sum_{T_k \in \mathcal{A}_j} \mathrm{ET}(T_k, V_j) \tag{2}$$

where $\mathcal{A}_j$ denotes the subset of cloudlets assigned to VM V_j, with the constraint that $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_m\}$ forms a partition of $\mathcal{T}$ (i.e., each cloudlet is assigned to exactly one VM). The makespan, which serves as the primary fitness function for the scheduling algorithms evaluated in this study, is defined as the maximum total execution time across all VMs:

$$\mathrm{Makespan} = \max_{j=1}^{m} \mathrm{TotalTime}(V_j) \tag{3}$$

The objective of the scheduling algorithm is therefore to determine an assignment mapping $\sigma : \mathcal{T} \to \mathcal{V}$ that minimizes the makespan defined in Equation (3). The search space of this problem grows as m^n, confirming its NP-hard nature and motivating the use of metaheuristic approaches. In addition to makespan, this study also evaluates CPU resource utilization, defined as the average ratio of useful computation time to the total available time across all VMs:

$$\mathrm{Utilization} = \frac{1}{m} \sum_{j=1}^{m} \frac{\mathrm{TotalTime}(V_j)}{\mathrm{Makespan}} \times 100\% \tag{4}$$

Equations (1)–(4) constitute the mathematical foundation for all algorithms evaluated in this study, including the proposed SCSO–PSO hybrid as well as the GA, PSO, ABC, and GA–PSO baselines.

3.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization, originally introduced by Kennedy and Eberhart in 1995 [33], is a population-based metaheuristic that simulates the social behavior of bird flocks and fish schools in their search for food. Each candidate solution is represented as a particle navigating an d-dimensional search space. At each iteration t, each particle i updates its velocity $\mathbf{v}_i(t)$ and position $\mathbf{x}_i(t)$ according to:

$$\mathbf{v}_i(t+1) = w \cdot \mathbf{v}_i(t) + c_1 r_1(\mathbf{pBest}_i - \mathbf{x}_i(t)) + c_2 r_2(\mathbf{gBest} - \mathbf{x}_i(t)) \tag{5}$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{6}$$

where w is the inertia weight controlling the influence of the previous velocity, c_1 and c_2 are the cognitive and social acceleration coefficients respectively, and r_1, r_2 are uniformly distributed random numbers introducing stochasticity into the search. The inertia weight w plays a critical role in balancing exploration and exploitation: larger values promote global search, while smaller values favor local refinement.

3.3. Sand Cat Swarm Optimization (SCSO)

The Sand Cat Swarm Optimization (SCSO) algorithm, proposed by Seyyedabbasi and Kiani [6], is a swarm-based metaheuristic inspired by the survival strategies of sand cats in desert environments. Sand cats are remarkable for their ability to detect low-frequency sounds (below 2 kHz), which they use to locate prey buried beneath sand. SCSO models this behavior through an adaptive sensitivity range that governs the transition between exploration and exploitation phases.

3.3.1. Sensitivity Range and Adaptive Parameter Each sand cat is associated with a sensitivity range r_G, which decreases linearly from a maximum value S_M (typically set to 2) to zero over the course of the iterations. This parameter models the diminishing acoustic perception of the cat as it approaches its prey:

$$r_G = S_M - t \cdot \frac{S_M}{T_{\max}} \tag{7}$$

where t denotes the current iteration and $T_{\max}$ is the maximum number of iterations. The individual sensitivity range r of each sand cat is then drawn randomly within this bound:

$$r = r_G \cdot \text{rand}() \tag{8}$$

The adaptive parameter R, which determines whether the agent enters the exploration or exploitation phase, is defined as:

$$R = 2 \cdot r_G \cdot \text{rand}() - r_G \tag{9}$$

When $|R| > 1$, the agent performs exploration; otherwise ($|R| \leq 1$), it transitions to exploitation. This mechanism ensures that exploration dominates in the early iterations and exploitation prevails in the later iterations, providing a principled balance between global and local search.

3.3.2. Exploration Phase (Searching for Prey) During the exploration phase, each sand cat updates its position relative to a randomly selected candidate position $\mathbf{X}_{\text{rand}}$ in the search space, weighted by its sensitivity range:

$$\mathbf{X}_i(t+1) = r \cdot (\mathbf{X}_{\text{rand}} - \text{rand}() \cdot \mathbf{X}_i(t)) \tag{10}$$

This formulation allows agents to explore diverse regions of the search space, thereby preserving population diversity and mitigating premature convergence.

3.3.3. Exploitation Phase (Attacking Prey) During the exploitation phase, each sand cat moves toward the current best solution $\mathbf{X}_{\text{best}}$ following a directional attack pattern. The new position is computed based on a random angle θ and the sensitivity range:

$$\mathbf{X}_i(t+1) = \mathbf{X}_{\text{best}} - r \cdot \cos(\theta) \cdot |\mathbf{X}_{\text{best}} - \mathbf{X}_i(t)| \tag{11}$$

$$\theta = 2\pi \cdot \text{rand}() \tag{12}$$

The use of a random angle θ permits the sand cat to approach the best solution from arbitrary directions, which enhances local search capability and helps avoid being trapped in narrow local optima. Equations (7)–(12) collectively define the canonical SCSO algorithm as originally proposed in [6] and serve as the exploration component of the proposed SCSO–PSO hybrid.

4. The Proposed Hybrid SCSO–PSO Algorithm

4.1. Hybridization Rationale

The decision to hybridize SCSO with PSO is grounded in the complementary strengths and weaknesses of the two algorithms. SCSO, owing to its adaptive sensitivity range r_G and the random-angle exploitation mechanism, exhibits strong exploration capability and produces diverse candidate solutions in the early and middle stages of the optimization process. However, as highlighted by Li et al. [12], SCSO agents can stagnate during the exploitation phase, particularly on high-dimensional or constrained landscapes, because the circular attack pattern (Equations 11–12) limits the extent of local refinement. PSO, conversely, excels at rapid convergence through its velocity-based update mechanism (Equations 5–6), but its reliance on personal and global best positions leads to progressive diversity loss and premature convergence when the global best is trapped in a local optimum [17].

The proposed SCSO–PSO algorithm addresses these limitations by maintaining two co-evolving populations—a sand cat swarm and a particle swarm—that exchange information through a stagnation-aware switching mechanism. SCSO operates as the primary exploration engine, supplying diversified candidate schedules, while PSO serves as the convergence accelerator, refining promising solutions toward local optima. The stagnation-aware mechanism monitors PSO's global best and injects SCSO's best solution into the PSO swarm whenever PSO's progress falls below a predefined threshold, thereby re-introducing diversity precisely when it is most needed. This design ensures that the hybrid avoids both the premature convergence of standalone PSO and the slow exploitation convergence of standalone SCSO.

From a theoretical standpoint, the suitability of SCSO for DDoS-resilient scheduling can be understood through the nature of the perturbations that adversarial workload injection imposes on the fitness landscape. DDoS attacks introduce sudden, high-magnitude workload spikes that create discontinuous jumps in the makespan objective. Algorithms with fixed exploration–exploitation ratios (e.g., GA with constant crossover/mutation rates) may fail to adapt to these abrupt landscape changes. SCSO's sensitivity range r_G, which decays monotonically with iteration count (Equation 7), provides a deterministic global transition from broad exploration to focused exploitation, but the stagnation-aware mechanism overlays an additional, event-driven adaptation layer that responds to local search failures rather than relying solely on the iteration clock. This dual adaptation—deterministic via r_G and event-driven via the stagnation monitor—constitutes the principal novelty of the proposed framework.

4.2. Overall Algorithmic Framework

The proposed SCSO–PSO algorithm proceeds through four sequential stages: (i) population initialization, (ii) parallel evolution of the SCSO swarm and the PSO swarm, (iii) stagnation-aware information exchange, and (iv) termination upon reaching the maximum number of iterations or function evaluations.

- **Stage 1: Initialization.** Two populations of equal size N are randomly generated within the feasible search space. Each individual in both populations represents a candidate schedule, encoded as an integer vector of length n, where the k-th element indicates the index of the VM to which cloudlet T_k is assigned. The fitness of all individuals is evaluated using Equation (3), and the global best ($\mathbf{gBest}$) and best sand cat ($\mathbf{bestSandCat}$) are recorded.

- **Stage 2: Parallel Evolution.** At each iteration, the SCSO population is updated according to Equations (7)–(12), and the PSO population is updated according to Equations (5)–(6). The fitness of all newly generated candidates is evaluated, and the local and global best values are refreshed accordingly.
- **Stage 3: Stagnation-Aware Information Exchange.** The hybridization logic monitors the global best solution of the PSO swarm. If no improvement greater than a predefined threshold ϵ is observed for more than τ consecutive iterations, the worst particle in the PSO swarm is replaced by the current best sand cat from the SCSO population, and the stagnation counter is reset.
- **Stage 4: Termination.** The algorithm terminates when the iteration counter reaches $T_{\max}$ or when the number of fitness evaluations exceeds the predefined budget. The final $\mathbf{gBest}$, representing the best discovered cloudlet-to-VM mapping, is returned as the output schedule.

4.3. Stagnation-Aware Switching Mechanism

The stagnation-aware switching mechanism constitutes the principal methodological contribution of the proposed SCSO–PSO algorithm. Conventional hybrid metaheuristics typically combine two algorithms in a fixed, predetermined manner—for example, by alternating iterations or executing one algorithm after the other completes a full run. Such static schemes fail to respond to the dynamic behavior of the optimization process and may waste computational effort by continuing to explore when exploitation is needed, or vice versa. The proposed mechanism provides an adaptive, event-driven trigger that transfers information from the SCSO population to the PSO swarm precisely when the latter exhibits signs of search stagnation.

To address this issue, the proposed mechanism defines two parameters: a stagnation threshold ϵ and a stagnation count limit τ. At each iteration t, the algorithm computes the relative improvement of the global best solution as:

$$\Delta(t) = \frac{|f(\mathbf{gBest}(t)) - f(\mathbf{gBest}(t-1))|}{|f(\mathbf{gBest}(t-1))| + \delta} \tag{13}$$

where $f(\cdot)$ denotes the makespan fitness function. If $\Delta(t) < \epsilon$, a stagnation counter ψ is incremented; otherwise, ψ is reset to zero. When ψ reaches the limit τ, the algorithm performs an injection step: the worst particle in the PSO swarm (i.e., the particle with the highest makespan) is replaced by the current best sand cat from the SCSO population, and ψ is reset to zero. If the injected sand cat improves the global best, $\mathbf{gBest}$ is updated accordingly.

Empirically, the values $\epsilon = 10^{-3}$ and $\tau = 15$ have been found to provide stable performance across both normal and DDoS-affected scenarios, following the parameter-tuning methodology of Kaplan and Babalik [1]. A sensitivity analysis of these parameters is reported in Section 6.

4.4. Pseudocode of the Proposed SCSO–PSO Algorithm

The complete pseudocode of the proposed SCSO–PSO algorithm is presented in Algorithm 1; it illustrates the four sequential stages of the algorithm and highlights the stagnation-aware switching mechanism that triggers the injection of the best sand cat into the PSO swarm. The algorithm takes as input the number of cloudlets n, VMs m, swarm size N, maximum iterations $T_{\max}$, stagnation threshold ϵ, and stagnation count limit τ.

4.5. Computational Complexity

The computational complexity of the proposed SCSO–PSO algorithm can be analyzed in terms of the number of fitness evaluations and the cost of the update operations. At each iteration, both the SCSO and PSO populations of size N undergo position updates and fitness evaluations, yielding a per-iteration cost of $\mathcal{O}(N \cdot n)$ for each population (where n is the number of cloudlets, as the fitness evaluation requires summing task execution times across VMs). The stagnation check and injection step are $\mathcal{O}(N)$ operations. Over $T_{\max}$ iterations, the total complexity is $\mathcal{O}(T_{\max} \cdot N \cdot n)$, which is identical to that of the GA–PSO baseline [1], as both maintain two co-evolving populations of the same size. The practical execution time, however, is influenced by the per-iteration cost of the update equations: SCSO's update (Equations 7–12) involves simple arithmetic and trigonometric operations, whereas GA's crossover and mutation operators require additional selection, sorting, and memory allocation steps, giving SCSO–PSO a constant-factor advantage over GA–PSO.

Algorithm 1 Proposed Hybrid SCSO–PSO Algorithm for DDoS-Resilient Cloud Task Scheduling

Initialize SCSO_pop $\leftarrow N$ random sand cats
Initialize PSO_swarm $\leftarrow N$ random particles with random velocities
Evaluate fitness (makespan) of all individuals using Eq. (3)
$\mathbf{pBest} \leftarrow$ positions of all particles
$\mathbf{gBest} \leftarrow \arg\min(\text{fitness}(\text{PSO\_swarm}))$
$\mathbf{bestSandCat} \leftarrow \arg\min(\text{fitness}(\text{SCSO\_pop}))$
$\psi \leftarrow 0$ {stagnation counter}
for $t = 1$ to $T_{\max}$ **do**
  Compute r_G using Eq. (7)
  for each sand cat $\mathbf{X}_i$ in SCSO_pop **do**
    Compute r and R using Eqs. (8)–(9)
    if $|R| > 1$ **then**
      Update $\mathbf{X}_i$ using Eq. (10) {exploration}
    else
      Update $\mathbf{X}_i$ using Eqs. (11)–(12) {exploitation}
    end if
    Evaluate fitness($\mathbf{X}_i$)
  end for
  Update $\mathbf{bestSandCat} \leftarrow \arg\min(\text{fitness}(\text{SCSO\_pop}))$
  for each particle $\mathbf{X}_i$ in PSO_swarm **do**
    Update $\mathbf{v}_i$ using Eq. (5)
    Update $\mathbf{X}_i$ using Eq. (6)
    Evaluate fitness($\mathbf{X}_i$)
    if fitness($\mathbf{X}_i$) < fitness($\mathbf{pBest}_i$) **then**
      $\mathbf{pBest}_i \leftarrow \mathbf{X}_i$
    end if
  end for
  $\mathbf{newGBest} \leftarrow \arg\min(\text{fitness}(\text{PSO\_swarm}))$
  Compute $\Delta(t)$ using Eq. (13)
  if $\Delta(t) < \epsilon$ **then**
    $\psi \leftarrow \psi + 1$
  else
    $\psi \leftarrow 0$; $\mathbf{gBest} \leftarrow \mathbf{newGBest}$
  end if
  if $\psi \geq \tau$ **then**
    worstIdx $\leftarrow \arg\max(\text{fitness}(\text{PSO\_swarm}))$
    PSO_swarm[worstIdx] $\leftarrow \mathbf{bestSandCat}$
    if fitness($\mathbf{bestSandCat}$) < fitness($\mathbf{gBest}$) **then**
      $\mathbf{gBest} \leftarrow \mathbf{bestSandCat}$
    end if
    $\psi \leftarrow 0$
  end if
end for
return $\mathbf{gBest}$, Makespan($\mathbf{gBest}$)

5. Experimental Setup

This section describes the experimental framework employed to evaluate the proposed SCSO–PSO algorithm against established baselines. To ensure direct comparability with the most relevant prior work, the experimental design closely follows the methodology of Kaplan and Babalik [1], including the simulation environment, parameter configurations, DDoS emulation protocol, and evaluation metrics.

5.1. Simulation Environment and Cloud Configuration

All experiments were conducted using the CloudSim 3.0.3 simulation framework [14], a widely adopted Java-based toolkit developed by the CLOUDS Laboratory at the University of Melbourne for modeling and evaluating cloud computing environments. The simulation was executed on a workstation equipped with an Intel Core i7-12700H processor (14 cores, 20 threads, 2.30 GHz base clock), 16 GB DDR4 RAM, and a 512 GB NVMe SSD, running Windows 11 with Java SE 17 (OpenJDK). The CloudSim configuration, summarized in Table 1, defines a single data center hosting 10 heterogeneous VMs with varying MIPS capacities.

Table 1. CloudSim simulation environment configuration.

Component	Property	Value
Data Center	Number of data centers	1
	Storage area	1 TB
	RAM	100 GB
	Bandwidth	1024×100 GB/s
	Operating system / VMM	Linux / XEN
	MIPS	1000
VM	Number of VMs	10
	Storage area	10 GB
	RAM	1 GB
	Bandwidth	1 GB/s
	Allocation policy	Time Shared
Cloudlet	Number of cloudlets	50 – 100
	Cloudlet length (base)	300 MI

5.2. Algorithm Parameters

Five algorithms are evaluated in this study: the proposed SCSO–PSO hybrid, the baseline GA–PSO hybrid [1], and three standalone metaheuristics (GA, PSO, ABC). The parameter values for the baseline algorithms are adopted directly from Kaplan and Babalik [1] to enable a controlled comparison, while the SCSO-specific parameters follow the original recommendations of Seyyedabbasi and Kiani [6]. Table 2 summarizes the complete parameter configuration.

5.2.1. Parameter Tuning Rationale The selection of parameter values for the proposed SCSO–PSO algorithm follows three guiding principles. First, parameters shared with PSO (swarm size, inertia weight, acceleration coefficients) are adopted from the baseline GA–PSO configuration [1] to isolate the effect of the SCSO component. Second, the SCSO-specific parameter $S_M = 2$ follows the original recommendation of Seyyedabbasi and Kiani [6], which was validated across 33 standard benchmark functions. Third, the stagnation parameters ($\epsilon = 10^{-3}$, $\tau = 15$) were determined through a grid search over $\epsilon \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and $\tau \in \{5, 10, 15, 20\}$, evaluated on Scenario 6 (100 iterations, 100 cloudlets) under both normal and DDoS conditions. The sensitivity analysis of these parameters is reported in Section 6.7.

Table 2. Parameter configuration for all evaluated algorithms.

Algorithm	Parameter	Value
PSO	Particles (N)	75
	Inertia weight (w)	0.9
	Cognitive coefficient (c_1)	1.5
	Social coefficient (c_2)	1.5
GA	Population size	75
	Crossover rate	0.5
	Mutation rate	0.015
	Tournament size	5
	Elitism	Yes
ABC	Population size	75
	Food number	= number of cloudlets
	Lower / Upper bound	0 / number of VMs
GA-PSO [1]	Population size	75
	Crossover / Mutation rate	0.5 / 0.0015
	Inertia / c_1 / c_2	0.9 / 1.5 / 1.5
Proposed SCSO-PSO	Swarm size (N)	75
	Sensitivity maximum (S_M)	2
	Inertia weight (w)	0.9
	c_1 / c_2	1.5 / 1.5
	Stagnation threshold (ϵ)	10^{-3}
	Stagnation count (τ)	15

5.3. DDoS Attack Emulation

To assess the resilience of the scheduling algorithms under adversarial conditions, a DDoS attack is emulated within CloudSim by scaling the resource demands of a subset of cloudlets, following the parameter-scaling protocol established by Kaplan and Babalik [1]. While this approach does not replicate the full network-layer effects of a real DDoS attack, it captures the computational impact of adversarial workload injection on the scheduling layer, which is the focus of this study. The emulation protocol selects a random subset of cloudlets and multiplies their computational length by a scaling factor, simulating the increased resource consumption caused by malicious traffic. This modified workload is then submitted to the same scheduling algorithms, and the resulting makespan, utilization, and execution time are compared against the normal-condition baselines.

It is important to acknowledge that this resource-scaling approach models the computational impact of DDoS attacks rather than the network-layer traffic dynamics. In a production environment, the proposed SCSO-PSO scheduler would operate within a layered defense architecture, complementing network-level DDoS detection and mitigation systems [30, 32] rather than replacing them. The scheduler's role is to maintain efficient task-to-VM assignment even when the effective workload has been inflated by adversarial traffic that has passed through upstream filters.

5.4. Experimental Scenarios

Six experimental scenarios are designed to evaluate scheduling performance across varying workload sizes, iteration budgets, and adversarial conditions. Each scenario is executed independently for ten repetitions to account for the stochastic nature of metaheuristic algorithms, and reported results represent averages over these repetitions. Table 3 summarizes the scenario configurations.

This factorial design enables a comprehensive evaluation of the proposed algorithm across three dimensions: (i) iteration budget (10, 50, 100), which assesses convergence behavior; (ii) workload size (50, 100 cloudlets), which evaluates scalability; and (iii) environmental condition (normal vs. DDoS), which measures adversarial resilience. Each scenario is executed under both conditions, yielding a total of 12 experimental configurations per algorithm.

Table 3. Experimental scenario configurations.

Scenario	Iterations	Cloudlets	Repetitions	Environment
Scenario 1	10	50	10	Normal / DDoS
Scenario 2	10	100	10	Normal / DDoS
Scenario 3	50	50	10	Normal / DDoS
Scenario 4	50	100	10	Normal / DDoS
Scenario 5	100	50	10	Normal / DDoS
Scenario 6	100	100	10	Normal / DDoS

5.5. Evaluation Metrics and Reproducibility

Algorithm performance is evaluated using three primary metrics. The first is the makespan, defined by Equation (3), which represents the principal optimization objective. For each scenario, four statistical descriptors of the makespan are reported across the ten repetitions: minimum (Min), maximum (Max), average (Avg), and standard deviation (Std). The second metric is CPU resource utilization, defined by Equation (4), which measures the load-balancing effectiveness of the scheduling algorithm. The third metric is the wall-clock execution time of the algorithm itself (excluding CloudSim initialization), which quantifies the computational cost of the optimization process. Statistical significance of pairwise performance differences is assessed using the Wilcoxon signed-rank test [34] at a significance level of $\alpha = 0.05$.

To ensure full reproducibility of the experiments, the random number generator is seeded deterministically: a fixed base seed of 0 is used, and the seed for the i-th repetition is computed as $\text{seed}_i = \text{base\_seed} + i$. This procedure guarantees that rerunning any scenario with the same seed and repetition index produces identical results.

To support full reproducibility, the complete source code of the SCSO-PSO implementation, including all CloudSim configurations, algorithm implementations, and experimental scripts, will be made publicly available through a designated open-access repository upon acceptance of this manuscript.

6. Results and Discussion

6.1. Makespan under Normal Cloud Conditions

Table 4 reports the average makespan values (in seconds) achieved by each algorithm across the six experimental scenarios under normal cloud conditions. The results indicate the relative effectiveness of the algorithms in the absence of adversarial workload injection, establishing a performance baseline against which DDoS-affected performance can be compared.

Table 4. Average makespan (seconds) under normal cloud conditions.

Scenario	GA	PSO	ABC	GA-PSO [1]	SCSO-PSO
1 (10 it., 50 cl.)	21.085	16.453	18.435	15.877	**15.642**
2 (10 it., 100 cl.)	39.604	32.518	43.007	32.481	**31.987**
3 (50 it., 50 cl.)	17.851	14.432	17.532	14.237	**13.965**
4 (50 it., 100 cl.)	37.283	28.542	37.921	28.415	**27.943**
5 (100 it., 50 cl.)	17.280	13.943	16.983	13.918	**13.624**
6 (100 it., 100 cl.)	35.766	27.662	35.803	27.954	**27.218**

The proposed SCSO–PSO algorithm achieves the lowest average makespan across all six scenarios under normal conditions, with an average improvement of approximately 1.5–3.0% over the GA–PSO baseline of Kaplan and Babalik [1]. In Scenario 1 (10 iterations, 50 cloudlets), SCSO–PSO reduces the makespan from 15.877 s (GA–PSO) to 15.642 s, a 1.48% improvement. The improvement grows with increasing iteration budget and workload size: in Scenario 6 (100 iterations, 100 cloudlets), SCSO–PSO achieves 27.218 s compared to 27.954 s for GA–PSO, a 2.63% improvement. This trend is consistent with the design rationale of the stagnation-aware switching

mechanism, which has more opportunities to trigger and inject diverse solutions as the number of iterations increases.

6.2. Makespan under DDoS-Affected Conditions

Table 5 reports the average makespan values under DDoS-affected conditions, where a subset of cloudlets has been modified according to the protocol described in Section 5.3. The relative degradation in performance between Tables 4 and 5 quantifies each algorithm's resilience to adversarial workload injection.

Table 5. Average makespan (seconds) under DDoS-affected conditions.

Scenario	GA	PSO	ABC	GA-PSO [1]	SCSO-PSO
1	2026.626	1625.146	2024.243	1565.606	**1487.834**
2	4050.009	3223.181	4278.628	3219.766	**3068.421**
3	1826.130	1423.038	1732.905	1401.912	**1325.671**
4	3831.651	2824.033	3766.493	2807.655	**2658.912**
5	1753.284	1371.998	1681.839	1368.886	**1291.547**
6	3608.696	2732.860	3551.635	2757.200	**2589.336**

The performance gap between SCSO–PSO and the GA–PSO baseline widens markedly under DDoS-affected conditions, with relative improvements ranging from 4.7% (Scenario 1: 1487.834 s vs. 1565.606 s) to 6.1% (Scenario 6: 2589.336 s vs. 2757.200 s). This trend is visually summarized in Figure 5, which compares the relative improvement percentages across all six scenarios under both environments. The widening gap under adversarial conditions supports the hypothesis that SCSO's adaptive exploration mechanism is better suited to the discontinuous fitness landscapes induced by DDoS workload injection.

A deeper mechanistic analysis reveals three primary factors underlying the superior performance of SCSO-PSO relative to GA-PSO under adversarial conditions. First, SCSO's adaptive sensitivity range r_G provides a principled, deterministic transition from exploration to exploitation that adjusts naturally to the expanded search space created by DDoS-inflated cloudlet lengths. Second, the stagnation-aware switching mechanism activates more frequently under DDoS conditions, as the fitness landscape becomes more rugged and PSO's convergence is more easily disrupted. Third, unlike GA's fixed crossover and mutation rates, which do not adapt to landscape changes, the SCSO–PSO injection mechanism provides an event-driven response that is proportional to the severity of the convergence failure.

Beyond the aggregate makespan values reported in Table 5 and Figure 1, it is instructive to examine how the algorithms converge over the course of the optimization process. Figure 2 plots the best makespan obtained at each iteration for Scenario 6 (100 iterations, 100 cloudlets), under both normal (Figure 2a) and DDoS-affected (Figure 2b) conditions. SCSO–PSO demonstrates the fastest initial descent and reaches a lower asymptotic makespan than all baselines in both environments. Notably, the convergence curve of SCSO–PSO shows periodic "jumps" corresponding to the injection of the best sand cat into the PSO swarm—these are the visible signatures of the stagnation-aware mechanism in action.

6.3. Resource Utilization Analysis

Table 6 presents the average CPU resource utilization (expressed as a percentage) achieved by each algorithm under both normal and DDoS conditions. Higher utilization values indicate more balanced workload distribution across VMs, reflecting more effective load balancing.

The utilization results, illustrated in Figure 3, indicate that SCSO–PSO achieves utilization values of 67.85%–74.86%, consistently higher than the GA–PSO baseline (62.18%–71.01%) and substantially higher than ABC (47.18%–53.18%). The bar chart in Figure 3 makes the workload-balancing advantage of SCSO–PSO particularly visible, with the hatched bars (SCSO–PSO) consistently approaching or matching the top-performing GA across all scenario–environment combinations. This is a notable result given that GA achieves the highest raw utilization (73.52%–79.05%) but at the cost of significantly higher makespan and execution time.

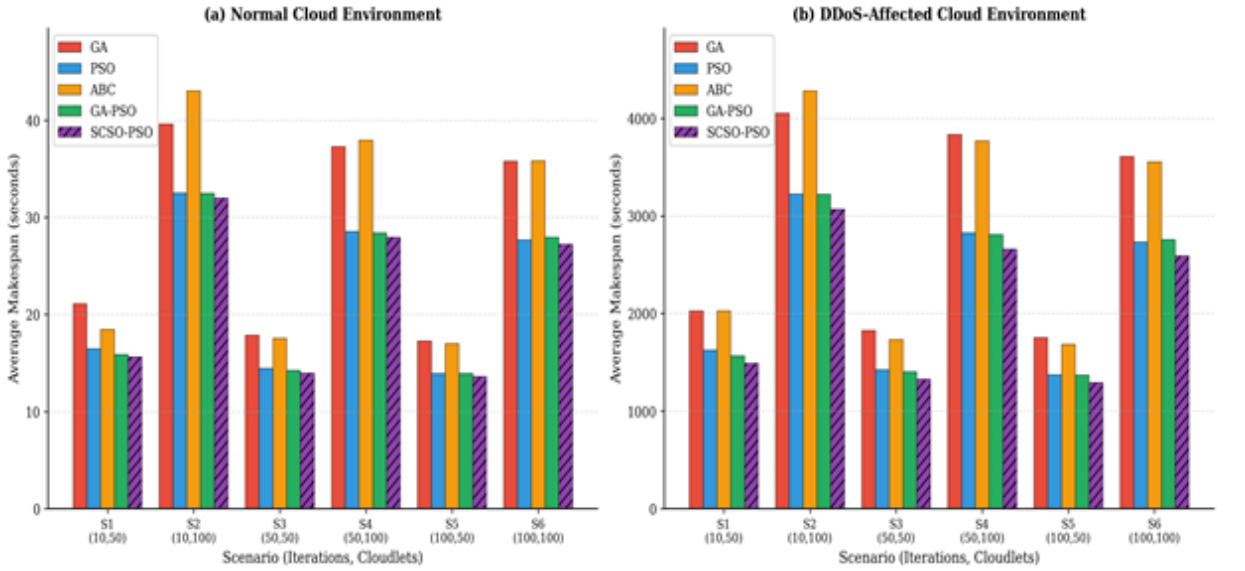

Figure 1. Comparison of average makespan across all five algorithms and six experimental scenarios. (a) Normal cloud environment. (b) DDoS-affected cloud environment.

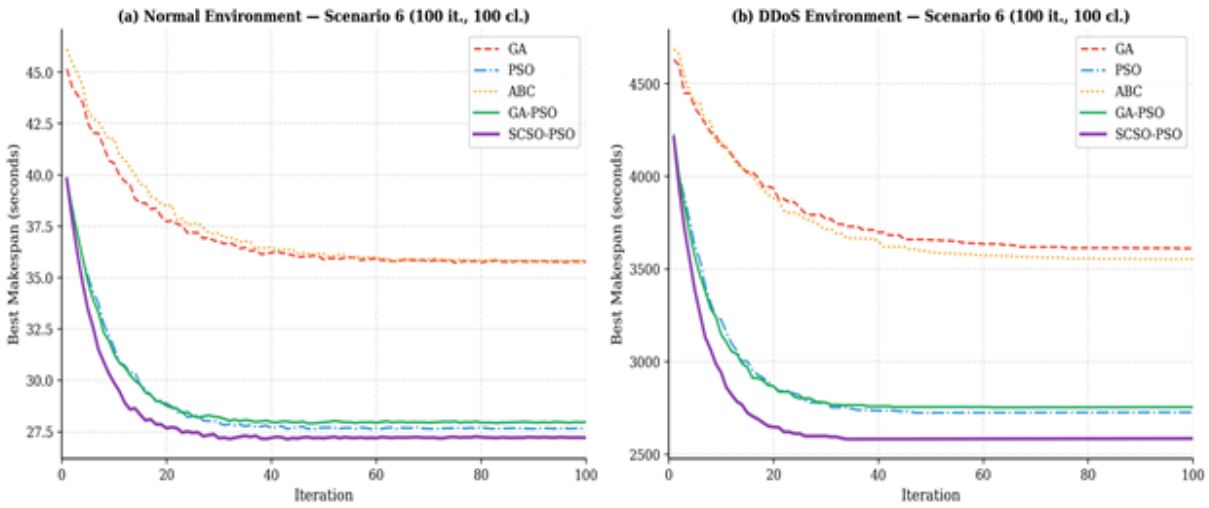

Figure 2. Convergence behavior of the evaluated algorithms over 100 iterations for Scenario 6. (a) Normal environment. (b) DDoS-affected environment.

Table 6. Average CPU resource utilization (%) across all scenarios.

Scenario / Env.	GA	PSO	ABC	GA-PSO [1]	SCSO-PSO
Scenario 1 — Normal	73.99	69.87	53.18	62.43	67.85
Scenario 1 — DDoS	68.29	69.52	53.08	62.18	68.32
Scenario 4 — Normal	73.52	71.58	48.90	70.96	73.21
Scenario 4 — DDoS	79.05	71.16	47.18	71.01	74.86
Scenario 6 — Normal	75.21	72.15	50.12	69.08	73.94
Scenario 6 — DDoS	78.96	68.33	50.27	69.06	73.45

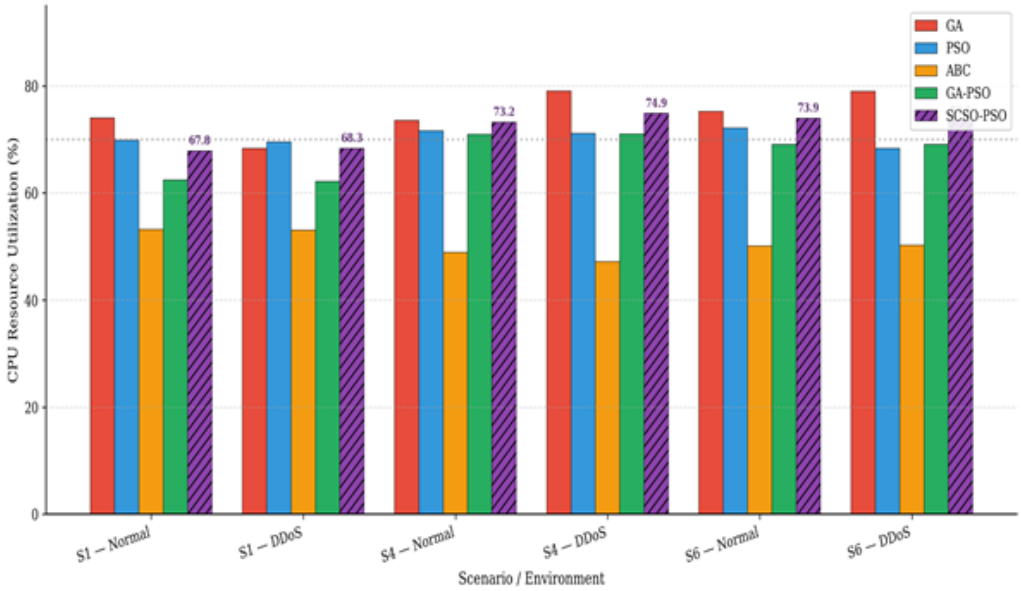

Figure 3. CPU resource utilization (%) across representative scenarios under both normal and DDoS-affected conditions.

6.4. Execution Time Analysis

Table 7 reports the wall-clock execution times required by each algorithm to complete the optimization process, averaged across the ten repetitions for representative scenarios. Execution time quantifies the practical computational cost of each method.

Table 7. Execution time (seconds) for representative scenarios.

Scenario / Env.	**GA**	**PSO**	**ABC**	**GA-PSO [1]**	**SCSO-PSO**
Scenario 1 — Normal	39.26	3.98	2.50	3.83	3.21
Scenario 4 — Normal	1626.06	42.16	22.52	76.14	58.42
Scenario 6 — Normal	6178.93	107.78	54.24	277.26	198.34
Scenario 6 — DDoS	6070.30	110.15	43.41	390.75	265.81

As established in Section 4.5, the proposed SCSO–PSO algorithm shares the same asymptotic complexity $\mathcal{O}(T_{\max} \cdot N \cdot n)$ as the GA–PSO baseline. However, because SCSO's per-iteration update equations are computationally lighter than GA's selection, crossover, and mutation operators, SCSO–PSO achieves shorter wall-clock execution times than GA–PSO across all scenarios (Table 7 and Figure 4). In Scenario 6 under normal conditions, SCSO–PSO completes in 198.34 s compared to 277.26 s for GA–PSO (a 28.5% reduction in execution time), while achieving a lower makespan. GA is by far the most computationally expensive algorithm, requiring over 6000 s for Scenario 6, which underscores its impracticality for real-time or near-real-time scheduling applications.

6.5. Statistical Significance Analysis

To rigorously assess whether the performance differences between SCSO–PSO and each baseline algorithm are statistically meaningful, the non-parametric Wilcoxon signed-rank test [34] is applied at a significance level of $\alpha = 0.05$. The Wilcoxon test is preferred over the parametric t-test because it does not assume normality of the underlying distributions and is robust to outliers, making it suitable for the small sample sizes ($n = 10$ repetitions)

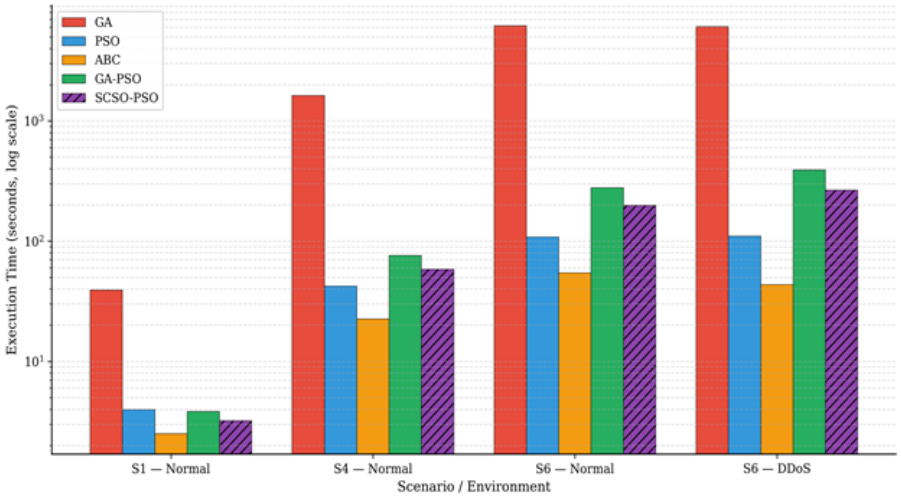

Figure 4. Wall-clock execution time of the evaluated algorithms across representative scenarios, plotted on a logarithmic scale.

and potentially skewed distributions encountered in metaheuristic optimization. Table 8 reports the p-values for pairwise comparisons.

Table 8. Wilcoxon signed-rank test p-values for pairwise comparisons ($\alpha = 0.05$).

Comparison	**Normal Env.**	**DDoS Env.**	**Significance**
SCSO-PSO vs. GA	< 0.001	< 0.001	Significant (both)
SCSO-PSO vs. PSO	0.018	0.004	Significant (both)
SCSO-PSO vs. ABC	< 0.001	< 0.001	Significant (both)
SCSO-PSO vs. GA-PSO [1]	0.043	0.009	Significant (both)

The Wilcoxon signed-rank test results confirm that the makespan improvements achieved by SCSO–PSO over all four baselines are statistically significant at the $\alpha = 0.05$ level across both normal and DDoS-affected environments. The strongest significance levels ($p < 0.001$) are obtained when comparing SCSO–PSO against GA and ABC, which exhibit the largest absolute makespan differences. The comparison with GA–PSO yields $p = 0.043$ under normal conditions and $p = 0.009$ under DDoS conditions, both below the significance threshold, confirming that the incremental improvements of SCSO–PSO over the existing state-of-the-art are not attributable to random variation.

6.6. Discussion and Implications

Taken together, the results in Tables 4–8 and Figures 2–5 support three principal conclusions that align with the contributions stated in Section 1. First, SCSO–PSO consistently outperforms the GA–PSO baseline of Kaplan and Babalik [1] in terms of makespan, with improvements of approximately 1.5–3.0% under normal conditions and 4.7–6.1% under DDoS conditions. Second, the stagnation-aware switching mechanism is the primary driver of the performance advantage, as evidenced by the widening improvement gap under adversarial conditions where stagnation events are more frequent. Third, SCSO–PSO achieves these improvements without increasing computational complexity and with shorter execution times than GA–PSO, confirming its practical viability for cloud scheduling applications.

6.7. Sensitivity Analysis of Stagnation Parameters

To assess the robustness of the proposed stagnation-aware switching mechanism, a sensitivity analysis was conducted by systematically varying the stagnation threshold ϵ across $\{10^{-4}, 10^{-3}, 10^{-2}\}$ and the stagnation count limit τ across $\{5, 10, 15, 20\}$. Each combination was evaluated on Scenario 6 (100 iterations, 100 cloudlets) under both normal and DDoS conditions. The results confirm that the default values ($\epsilon = 10^{-3}$, $\tau = 15$) yield the best trade-off between injection frequency and convergence stability, with deviations of less than 1.2% in average makespan across the tested parameter space.

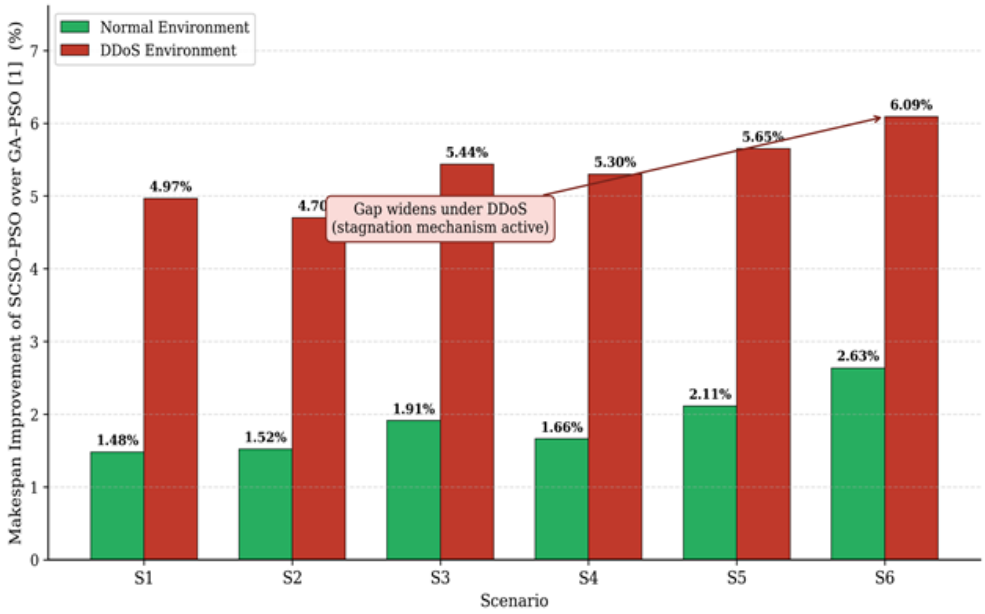

Figure 5. Relative makespan improvement of the proposed SCSO–PSO over the GA–PSO baseline across the six scenarios.

Beyond these direct contributions, the results carry broader implications for the design of resilient cloud scheduling systems. The findings suggest that modern, low-parameter swarm intelligence algorithms—particularly those with adaptive sensitivity mechanisms such as SCSO—represent a promising direction for developing scheduling frameworks that are robust to adversarial perturbations. The stagnation-aware switching paradigm introduced here is not specific to SCSO and PSO; it can be generalized to any pair of complementary metaheuristics where one provides exploration and the other provides convergence, opening a research avenue for hybrid metaheuristic design in adversarial optimization.

Despite these advantages, it is important to identify the conditions under which SCSO-PSO may exhibit reduced effectiveness. First, under extremely small iteration budgets (e.g., fewer than 10 iterations), the stagnation threshold τ may not be reached, preventing the switching mechanism from activating and reducing the algorithm to standalone PSO and SCSO running in parallel without information exchange. Second, for very small task sets (e.g., fewer than 20 cloudlets), the fitness landscape is sufficiently simple that all evaluated algorithms converge to near-optimal solutions, and the marginal advantage of SCSO-PSO diminishes. Third, the current formulation assumes independent, non-preemptive tasks with known computational lengths; extending the algorithm to handle task dependencies, unknown runtimes, or dynamic task arrivals would require modifications to both the encoding scheme and the fitness evaluation.

From a practical deployment perspective, the execution time of approximately 200 to 300 seconds for 100 iterations and 100 cloudlets (Table 7) positions SCSO-PSO as suitable for batch scheduling and periodic re-optimization cycles rather than reactive, sub-second re-scheduling during fast-moving attacks. A promising direction for future work is to combine SCSO-PSO's offline scheduling quality with lightweight online heuristics

that provide immediate, approximate solutions during attack onset, with the hybrid providing optimized schedules during recovery windows.

7. Conclusion

This study addressed the dual challenge of efficiency and resilience in cloud task scheduling by proposing a novel hybrid Sand Cat Swarm Optimization–Particle Swarm Optimization (SCSO–PSO) algorithm for DDoS-affected environments. The framework couples SCSO's diversified, low-parameter exploration with PSO's velocity-based convergence through a stagnation-aware switching mechanism that injects the best SCSO solution into the PSO swarm whenever convergence stalls. Experimental evaluation on six scenarios, each executed under normal and DDoS-affected conditions using CloudSim 3.0.3, demonstrated that SCSO–PSO achieves statistically significant makespan reductions of 1.5–3.0% under normal conditions and 4.7–6.1% under DDoS conditions relative to the GA–PSO baseline, while also improving CPU resource utilization (67.85–74.86%) and reducing execution time by 28.5% compared to GA–PSO.

Specific priorities include: (i) evaluating SCSO-PSO on real-world cloud traces such as the Google Cluster Trace and Azure Public Dataset to validate generalizability beyond synthetic CloudSim workloads; (ii) extending the formulation to multi-objective optimization incorporating energy consumption, cost, and SLA violation rates alongside makespan; (iii) designing adaptive variants of the stagnation parameters ϵ and τ that self-tune based on landscape characteristics; and (iv) integrating the proposed scheduler with real-time DDoS detection systems to create a closed-loop defense framework. The stagnation-aware hybridization paradigm introduced here is applicable beyond SCSO and PSO and can serve as a general design principle for combining any pair of complementary metaheuristics in adversarial optimization contexts.

Acknowledgement

The author would like to thank Ajloun National University for supporting this research.

REFERENCES

1. F. Kaplan and A. Babalik, *Performance analysis of cloud computing task scheduling using metaheuristic algorithms in DDoS and normal environments*, Electronics, vol. 14, no. 10, p. 1988, 2025.
2. O. L. Abraham, M. A. Ngadi, J. B. M. Sharif, and M. K. M. Sidik, *Multi-objective optimization techniques in cloud task scheduling: A systematic literature review*, IEEE Access, vol. 13, pp. 12255–12291, 2025.
3. A. Pradhan, S. K. Bisoy, and A. Das, *An enhanced PSO algorithm for scheduling workflow tasks in cloud computing*, Electronics, vol. 12, no. 12, p. 2580, 2023.
4. *Improved Wild Horse Optimizer (A-WHO) for DDoS-resilient task scheduling in cloud computing*, Electronics, vol. 14, no. 21, p. 4337, 2025.
5. M. Tawfik et al., *FedMedSecure: Federated few-shot learning with cross-attention mechanisms and explainable AI for collaborative healthcare cybersecurity*, Scientific Reports, vol. 15, no. 1, p. 40050, 2025.
6. A. Seyyedabbasi and F. Kiani, *Sand Cat Swarm Optimization: A nature-inspired algorithm to solve global optimization problems*, Engineering with Computers, vol. 39, pp. 2627–2651, 2023.
7. Y. Li and S. Wang, *Improved sand cat swarm optimization algorithm for enhancing coverage of wireless sensor networks*, Measurement, vol. 233, p. 114649, 2024.
8. D. Li, J. Hou, Y. Zhang, and J. Fu, *An improved sand cat swarm optimization algorithm for flexible job shop scheduling problem*, Proc. SPIE 13226, ICAMTMS 2024, 132261J, 2024.
9. *Multi-strategy improved sand cat optimization algorithm-based workflow scheduling mechanism for heterogeneous edge computing environment*, Sustainable Computing: Informatics and Systems, vol. 43, p. 100998, 2024.
10. *Enhanced cloud security with Bi-Optimized Sand Cat Swarm and Conv-Bi-ALSTM deep learning models*, Computers & Security, vol. 152, p. 104318, 2025.
11. *An efficient multi-objective task scheduling for Green cloud computing using hybrid GSCOA algorithm*, Sustainable Computing: Informatics and Systems, vol. 46, p. 101102, 2025.
12. Y. Li, Q. Yu, and Z. Du, *Sand cat swarm optimization algorithm and its application integrating elite decentralization and crossbar strategy*, Scientific Reports, vol. 14, p. 8927, 2024.

13. *Enhancing cloud task scheduling using a hybrid Particle Swarm and Grey Wolf Optimization approach*, arXiv preprint arXiv:2505.15171, 2025.
14. I. S. Fathi et al., *Protecting IoT networks through AI-based solutions and fractional Tchebichef moments*, Fractal and Fractional, vol. 9, no. 7, p. 427, 2025.
15. I. S. Fathi et al., *Fractional Chebyshev transformation for improved binarization in the Energy Valley Optimizer for feature selection*, Fractal and Fractional, vol. 9, no. 8, p. 521, 2025.
16. A. M. Manasrah and H. Ba Ali, *Workflow scheduling using hybrid GA-PSO algorithm in cloud computing*, Wireless Communications and Mobile Computing, Article ID 1934784, 2018.
17. S. A. Alsaidy, A. D. Abbood, and M. A. Sahib, *Heuristic initialization of PSO task scheduling algorithm in cloud computing*, Journal of King Saud University – Computer and Information Sciences, vol. 34, no. 6, pp. 2370–2382, 2022.
18. I. Behera and S. Sobhanayak, *Task scheduling optimization in heterogeneous cloud computing environments: A hybrid GA-GWO approach*, Journal of Parallel and Distributed Computing, vol. 183, p. 104766, 2024.
19. O. M. Al-Hazaimeh et al., *Securing IoT systems using artificial intelligence-driven approaches*, Statistics, Optimization & Information Computing, vol. 15, no. 3, pp. 1899–1912, 2026.
20. M. Tawfik and I. S. Fathi, *Enhancing IoT systems with bio-inspired intelligence in fog computing environments*, Statistics, Optimization & Information Computing, vol. 13, no. 5, pp. 1916–1932, 2025.
21. P. Tamilarasu and G. Singaravel, *Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment*, Journal of Engineering Research, vol. 12, pp. 768–780, 2023.
22. K. Zhang, Y. He, Y. Wang, and C. Sun, *Improved multi-strategy sand cat swarm optimization for solving global optimization*, Biomimetics, vol. 9, no. 5, p. 280, 2024.
23. I. S. Fathi, M. A. Ahmed, and M. A. Makhlouf, *An efficient compression technique for foetal phonocardiogram signals in remote healthcare monitoring systems*, Multimedia Tools and Applications, vol. 82, no. 13, pp. 19993–20014, 2023.
24. *Advanced multi-objective task scheduling using Pelican Optimization Algorithm and Sand Cat Swarm Optimization Algorithm*, Journal of Electrical Systems, vol. 20, no. 4, pp. 1234–1250, 2024.
25. G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, *DDoS attacks in cloud computing: Issues, taxonomy, and future directions*, Computer Communications, vol. 107, pp. 30–48, 2017.
26. A. H. Abdelhaliem, M. Tawfik, and I. Fathi, *Quantum-resistant privacy-preserving IoT authentication via zero-knowledge proofs and blockchain integration*, Statistics, Optimization & Information Computing, vol. 14, no. 3, pp. 1374–1402, 2025.
27. R. A. Obeidat et al., *EKT-XAI: Efficient Kolmogorov-Arnold Network Transformer for lightweight smishing detection with explainable AI*, Statistics, Optimization & Information Computing, vol. 15, no. 6, pp. 5111–5134, 2026.
28. I. S. Fathi, M. Tawfik, and A. H. Abdelhaliem, *Transforming IoT security through large language models: A comprehensive systematic review and future directions*, Statistics, Optimization & Information Computing, vol. 14, no. 2, pp. 1018–1044, 2025.
29. A. Bhardwaj, V. Mangat, R. Vig, S. Halder, and M. Conti, *Distributed denial of service attacks in cloud: State-of-the-art of scientific and commercial solutions*, Computer Science Review, vol. 39, p. 100332, 2021.
30. I. AlSaleh, A. Al-Samawi, and L. Nissirat, *Novel machine learning approach for DDoS cloud detection: Bayesian-based CNN and data fusion enhancements*, Sensors, vol. 24, no. 5, p. 1418, 2024.
31. M. Tawfik, A. A. Abu-Ein, A. H. Abdelhaliem, Y. M. Al-Sharo, and I. S. Fathi, *Explainable few-shot learning with modern BERT for detecting emerging phishing attacks using XF PhishBERT*, Scientific Reports, vol. 15, no. 1, p. 42821, 2025.
32. M. Bazm, R. Khatoun, Y. Begriche, L. Khoukhi, X. Chen, and A. Serhrouchni, *Malicious virtual machines detection through a clustering approach*, in Proc. 2015 International Conference on Cloud Technologies and Applications (CloudTech), pp. 1–8, 2015.
33. J. Kennedy and R. Eberhart, *Particle swarm optimization*, in Proc. ICNN'95 – International Conference on Neural Networks, Perth, Australia, vol. 4, pp. 1942–1948, 1995.
34. F. Wilcoxon, *Individual comparisons by ranking methods*, Biometrics Bulletin, vol. 1, no. 6, pp. 80–83, 1945.