electronics

MDPI

Article

A-WHO: Stagnation-Based Adaptive Metaheuristic for Cloud Task Scheduling Resilient to DDoS Attacks

Fatih Kaplan * and Ahmet Babalık

Department of Computer Engineering, Faculty of Computer and Information Sciences, Konya Technical University, 42250 Konya, Turkey; ababalik@ktun.edu.tr

* Correspondence: fatihkaplan42@gmail.com

Abstract

Task scheduling in cloud computing becomes significantly more challenging under Distributed Denial-of-Service (DDoS) attacks, as malicious workload injection disrupts resource availability and degrades Quality of Service (QoS). To address this issue, this study proposes an improved Wild Horse Optimizer (A-WHO) that incorporates a stagnation detection mechanism and a stagnation-driven adaptive leader perturbation strategy. The proposed mechanism dynamically applies a noise-guided perturbation into the stallion position only when no improvement is observed over a predefined threshold, enabling A-WHO to escape local optima without modifying the standard behavior of WHO in normal iterations. In addition, a DDoS-aware CloudSim environment is developed by generating attacker virtual machines and high-MI malicious cloudlets to emulate realistic resource exhaustion scenarios. A-WHO's performance is assessed using makespan, SLA violation rate, each of the QoS metrics, and energy consumption on normal and DDoS conditions. The experimental results indicate that A-WHO achieves the best absolute makespan and QoS metrics during an attack and competitive results under normal conditions. In comparison with the WHO, PSO, ABC, GA, SCA, and CSOA, the proposed approach demonstrates improved robustness and greater resilience to resource degradation attacks. These findings indicate that integrating stagnation-aware diversification into metaheuristic schedulers represents a promising direction for securing cloud task scheduling frameworks.

Keywords: cloud task scheduling; cloud computing; metaheuristic algorithms; DDoS attacks

check for updates

Academic Editor: George Angelos Papadopoulos

Received: 30 September 2025
Revised: 31 October 2025
Accepted: 3 November 2025
Published: 5 November 2025

Citation: Kaplan, F.; Babalık, A. A-WHO: Stagnation-Based Adaptive Metaheuristic for Cloud Task Scheduling Resilient to DDoS Attacks. *Electronics* **2025**, *14*, 4337. https://doi.org/10.3390/electronics14214337

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over the past decade, cloud computing has fundamentally transformed the information technology ecosystem, establishing itself as a cornerstone of modern computing infrastructures through its inherent advantages of flexible resource management, high scalability, and cost efficiency. Both enterprises and individual users increasingly rely on cloud platforms for critical processes such as data storage, computational power, and distributed service provisioning. Consequently, cloud computing offers more adaptive and scalable solutions compared to traditional computing paradigms.

Nevertheless, cloud computing is seen as an outgrowth of advances in distributed computing, parallel computing, and network computing, and the heterogeneous resource structure and highly dynamic workload characteristics of cloud environments make the task scheduling problem extremely complex, classifying it as an NP-hard optimization problem. The management and allocation of cloud resources have emerged as central

research areas [1]. Efficient allocation of tasks across virtual machines (VMs) with heterogeneous processing capabilities directly influences not only key performance indicators such as makespan (i.e., the total completion time of all tasks) but also affects Service Level Agreement (SLA) compliance, energy efficiency, and Quality of Service (QoS) guarantees. Inefficient scheduling decisions inevitably lead to prolonged execution times, increased energy consumption, and consequently, higher operational costs alongside degraded service quality for both providers and end users.

There is also an increase in certain cybersecurity threats, specifically Distributed Denial-of-Service (DDoS) attacks, which pose a significant risk to the availability and consistency of services in the cloud. DDoS attacks specifically target and overuse a cloud infrastructure's critical resources by monopolizing its network bandwidth, virtual CPU, and memory, preventing cloud systems from responding to legitimate user requests. In cloud systems, DDoS attacks, network bandwidth, virtual CPU, and memory, preventing cloud systems from responding to legitimate user requests. In cloud systems, DDoS attacks cause the schedulers to miss service level objectives and increase the makespan. The degraded performance observed in the system makes the deterministic scheduling approaches of yesteryears look less valuable. This pain points the need for resilient systems, which provide the means to adjust in real time. Defensive systems need optimization to survive in predatory settings. The need for optimization systems has become intrinsic and urgent.

Recent research highlights the increasing importance of maintaining planned resilience under DDoS attacks, focusing on cloud security. For example, to mitigate DDoS attacks against enterprise networks, a nonlinear mixed-integer program model was proposed, which is characterized by scheduling DDoS traffic with on-premises scrubbing at the local edge and on-demand scrubbing at remote clouds, with inputs of arbitrary dynamics and trade-offs between staying at suboptimal scrubbing locations and using different optimal locations with switching overhead [2]. SOE: A Multi-Objective Traffic Scheduling Engine for DDoS Mitigation with Isolation-Aware Optimization developed a multi-objective scheduling engine that formulates traffic allocation under attack as an optimization problem balancing benign traffic acceptance rate, malicious traffic interception rate, server load balancing, and malicious traffic isolation [3].

In this context, metaheuristic algorithms have attracted substantial attention due to their robustness, flexibility, and adaptive search capabilities in solving complex optimization problems. However, a majority of existing studies have primarily focused on normal operating conditions, overlooking the impact of adversarial scenarios such as DDoS attacks. Consequently, there is a lack of comprehensive investigations evaluating the resilience and performance trade-offs of scheduling algorithms across multiple metrics including makespan, SLA violations, energy consumption, and QoS under realistic attack conditions.

The current study aims to address this research gap in three primary ways:

1. Comparative Performance Analysis: We conduct systematic evaluations in the CloudSim environment by analyzing the performance of Genetic Algorithm (GA), Particle Swarm Optimizer (PSO), Artificial Bee Colony (ABC), Wild Horse Optimizer (WHO), Crow Search Optimizer Algorithm (CSOA), and Sine–Cosine (SCA) algorithms, considering a multidimensional set of performance indicators for attacks under DDoS conditions.
2. Enhanced WHO Algorithm: We introduce Adaptive-WHO (A-WHO), an improved version of the standard WHO algorithm, which integrates a stagnation-aware adaptive diversity mechanism. This adjustment significantly reduces the risk of premature convergence and, thus, improves scheduling effectiveness during normal and adversarial conditions.

3. Realistic DDoS Attack Modeling: We create a more realistic DDoS attack scenario by integrating malicious VMs and adversarial cloudlets, allowing a thorough assessment of the algorithms' resilience against service degradation and resource exhaustion in a realistic scenario.

This paper continues as follows. Section 2 describes the cloud task scheduling problem as well as the employed metaheuristic algorithms. In Section 3, the discussion centers on the experimental setup, DDoS attack modeling and performance metrics. The results under both normal and adversarial conditions, inclusive of the A-WHO statistical performance evaluation, are presented in Section 4. Key findings are summarized in Section 5 and possible future research directions are also provided.

2. Related Work

In cloud computing, efficiently handling the assignment of heterogeneous, dynamic resources while reducing the makespan, energy cost, SLA, and QoS is an NP-hard optimization problem. Due to the limited flexibility and premature convergence tendencies of deterministic algorithms, metaheuristic optimization techniques have gained prominence in recent years and have been widely adopted across the literature. Classical metaheuristic algorithms such as GA, PSO, ABC, and Differential Evolution (DE) have been extensively applied to cloud task scheduling and refined through various hybrid and adaptive approaches. A review of existing studies reveals that early research predominantly focused on single parameter optimization most frequently makespan whereas recent studies have evolved toward multi-parameter formulations incorporating energy efficiency, QoS, and cost considerations.

For instance, Abdel-Basset et al. proposed a Hybrid Differential Evolution (HDE)-based scheduler and using the CloudSim environment, demonstrated improvements on makespan and total execution time compared to several metaheuristics [4]. Similarly, Chandrashekar et al. propose an advanced metaheuristic method called the Hybrid Weighted Ant Colony Optimization (HWACO) algorithm, which aims to improve task scheduling and is evaluated against existing algorithms in terms of efficiency, makespan, and cost [5]. Amer et al. merged the Spider Monkey Optimization (SMO) and ACO algorithms to provide a multi-objective solution for addressing the scheduling problem [6]. Parthasaradi proposed a multi-objective strategy using Horse Herd–Squirrel Search Algorithm for solving scheduling problem in cloud computing [7]. Additionally, Abraham et al. performed a systematic review of metaheuristic-based scheduling algorithms, clustered methodological patterns, and outlined the history of cloud task scheduling simulations [8]. Cui et al. reduced completion time and execution cost significantly during performance on CloudSim by developing a QoS-concerned hybrid workflow scheduler by merging the Whale Optimization Algorithm with HEFT initialization [9]. In cloud environments, Nandagopal et al. improved energy efficiency and SLA compliance with a hybrid strategy that integrated Cuckoo Search and transformer-based task reallocation [10]. In combinatorial metaheuristics, Khaleel et al. achieved significant advancement in energy efficiency and cost efficiency [11]. Another QoS-driven approach, the IVPTS algorithm, showed better performance than ELBA and ERA and decreased makespan and energy usage while upholding high standards of QoS [12].

Recent work on metaheuristics applied to optimizing the cloud and web services has been concentrated on developing diversity preserving and stagnation-aware search strategies to opportunistically manage variable workload along with dynamically changing QoS. Complementing the classical swarm scheduling paradigms relationships built on GA, PSO, and ABC, recent adaptive scheduling frameworks built on Whale Optimization Algorithms (WOAs) have shown remarkable flexibility. Dahan [13] proposed a new method based on

MA-WOA (Multi-Agent Whale Optimization Algorithm) to solve the web service composition problem. The approach uses a multi-agent structure to improve the service selection process and combines multiple Quality of Service (QoS) criteria with a weighted fitness function. Similarly, Mangalampalli et al. [14] adapted WOA for trust-aware task scheduling in cloud environments, where the algorithm demonstrated competitive performance relative to other population-based optimizers.

Among emerging metaheuristic approaches, the WHO, proposed by Naruei and Keynia, the WHO (Wild Horse Optimizer), inspired by the social behavior of wild horses. The results demonstrate that WHO demonstrates competitive and successful performance in many scenarios [15]. Saravanan et al. achieved significant improvements in cloud computing platform assignment performance by integrating the Lévy Flight strategy into the original WHO algorithm and applying it to cloud mission scheduling [16]. However, these studies primarily focus on individual performance enhancements without conducting comprehensive evaluations under adversarial conditions or across multiple performance dimensions.

Despite the significant progress in multi-objective optimization under normal operating conditions, most of these studies have largely overlooked the impact of adversarial scenarios and dynamically fluctuating workloads. Specifically, research evaluating the effects of DDoS attacks known for dramatically increasing resource consumption on scheduling performance using multidimensional metrics remains scarce.

Although DDoS-aware scheduling has recently gained momentum, the number of studies focusing specifically on metaheuristic-based task scheduling under DDoS conditions remains limited. Kaplan and Babalık [17] compared several metaheuristic algorithms under both normal and adversarial settings, demonstrating the detrimental impact of DDoS attacks on makespan. However, there is still a lack of systematic research that evaluates scheduling resilience using a holistic performance framework encompassing makespan, SLA compliance, QoS, and energy consumption.

At the infrastructure level, recent work has investigated deep learning-based attack detection for threats such as DDoS and IoT and has shown that attention-enhancing sequence models (e.g., BiLSTM-based architectures) achieve promising detection accuracy in resource-constrained environments [18]. While such detection approaches are orthogonal to our planning-focused goal, early attack detection is crucial for resilient cloud operation, as it can reduce the surface area that task schedulers must withstand under adversarial load. Accordingly, our work positions A-WHO as a complementary resilience mechanism on the planning side, while deep learning-based detection frameworks can strengthen defenses on the monitoring side in future extensions.

To address these research gaps, this study proposes the Adaptive-WHO (A-WHO) algorithm, an enhanced variant of the original WHO algorithm incorporating a Stagnation-Based Adaptive Diversity Mechanism. The proposed approach enables a comprehensive multi-metric performance analysis under both normal and DDoS scenarios, thereby bridging critical gaps in the literature on WHO algorithm enhancements and attack-resilient task scheduling.

3. Materials and Method

This section provides a comprehensive description of the algorithms employed in the study, namely WHO, SCA, CSOA, PSO, GA, ABC, and the proposed A-WHO, as well as the experimental environment, datasets, performance metrics, and the DDoS attack simulation implemented on the CloudSim platform. All experiments were conducted on a DELL Latitude 3540 laptop equipped with a 13th-generation Intel Core i5-1335U processor (1.30 GHz), 8 GB RAM, and a 500 GB SSD, running the Windows 11 Pro operating system.

The simulations were carried out using CloudSim v3.0.3, an open-source, Java-based simulation framework developed by the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne. CloudSim has become one of the most widely used platforms in task scheduling research due to its capability to accurately model heterogeneous data center resources, VM-Cloudlet interactions, and workload distributions in a controlled yet realistic environment.

3.1. Task Scheduling Formulation and System Modeling

In cloud computing environments, the task scheduling problem aims to optimally assign a set of tasks (cloudlets) to multiple VMs with heterogeneous processing capacities. Due to the complexity of the combinatorial problem and the need to mitigate trade-off situations with intertwining objectives with flexibility, this problem is NP-hard. The problem focuses on minimizing the makespan, that is minimizing the total completion time, while simultaneously optimizing core performance thresholds consisting of SLA, energy consumption, and QoS parameters [19].

3.1.1. Makespan

In task scheduling problems, completion time is the total time required to complete all assigned tasks, along with an indication of the total completion time. It is calculated by subtracting the finish time for each task from the start time for each task. This approach allows for a comprehensive assessment of system performance in scenarios where tasks do not initiate at the same time, facilitating the analysis of total completion time across different resources.

3.1.2. Service Level Agreement (SLA) Compliance

Cloud services entail a contract signed by a provider and a user known as a Service Level Agreement (SLA), within which each task has a specific timeframe to be completed. The overtaking of this timeframe, therefore, must be completed late to be identified as a breach of the SLA. This approach diverges from traditional methods with individual task deadlines by employing a single, fixed time window for every task. Consequently, it equates to a single SLA standard and provides a comprehensive view of all scheduling algorithms within the same time framework, ensuring the disparity in performance can be solely attributed to the algorithms' adequacy, rather than varying deadlines [19].

3.1.3. Energy Consumption

In contemporary cloud computing architectures, the performance metrics that are determined on energy consumption have become indispensable. This critiques the operational costs, the environmental sustainability, and the objective of green computing. With the expansion of large data centers and the workloads that are more energy intensive, the integration of energy-aware scheduling policies has become a dominant concern in published research [19].

In this study, the total energy consumption is computed by considering both the active (busy) and idle power profiles of each virtual machine. The energy consumed by a given VM is calculated by multiplying the corresponding power consumption coefficients by the total time spent in each state.

All energy measurements are expressed in Joules (J), enabling a comparative analysis of the energy efficiency of different algorithms under varying workload intensities and adversarial conditions. By considering both execution and idle states, this approach provides a comprehensive perspective of the energy consumption profile of cloud infrastructures.

3.1.4. Quality of Service (QoS) Metrics

In this study, the evaluation of Quality of Service is based on Time-Based Quality of Service (TBQS) metrics, which measure system efficiency using time-oriented performance indicators derived from task execution and response times.

Specifically, two key components are considered:

- Execution Time (ET): The actual processing time during which a task is actively executed on the CPU.
- Response Time (RT): The total time elapsed from the moment a task is requested by the user until its complete execution.

It is important to note that SLA violation and energy consumption metrics are treated as independent performance criteria in this study. Hence, QoS evaluation focuses solely on time-based metrics.

Even though both SLA and QoS focus on time, they are different layers of performance. SLA violation rate is a rigid contractual performance indicator that measures whether a cloudlet crossed its deadline (violation or non-violation), while, on the other hand, QoS metrics (execution time, response time) offer a continuous measure of service performance. Consequently, even in the absence of an SLA violation, the QoS metrics assist in evaluating the relative effectiveness of the scheduling decisions.

3.1.5. Resource Utilization

In cloud environments, resource utilization (RU) is a core KPI, and a metric for evaluating the effectiveness of resource allocation, load balancing, and scheduling. RU metrics calculate the amount of time computing resources (like virtual machines) are engaged in task execution during a time window and compare that to the entire window length.

A high RU indicates that most of the allocated computational resources are working, suggesting that the scheduling algorithm is very effective. Low RU indicates the opposite; there are idle resources, which suggests load imbalance and poor scheduling.

In this study, RU is calculated by recording the busy time of each VM and normalizing it to the total simulation time.

3.2. Applied Metaheuristic Algorithms

This research utilized multiple metaheuristic optimization algorithms to solve the problem of task scheduling in cloud computing environments. In addition to the more common classical algorithms such as GA, PSO, and ABC, this work examined the WHO algorithm, which has seen less exploration in the area of cloud computing.

The WHO algorithm attempts to maximize exploration and exploitation of the search space using a novel approach that simulates the social behavior and hierarchal structure of age group-based horse herds. Although this algorithm has unique characteristics, the large number of parameters and complex multiple integrated components can cause the algorithm to prematurely converge and become stuck in local optima if the parameters are poorly set. In dynamic and adversarial cloud environments, these limitations can make a poorly designed algorithm perform unpredictably.

In this study, the stagnation-based adaptive diversity mechanism was employed to address weaknesses, thus leading to the development of the Adaptive-WHO (A-WHO) algorithm. As described, the stagnation-based adaptive diversity mechanism aids in the preservation of population diversity to avoid premature convergence while striking a balanced exploration trade-off. This is, to the best of our knowledge, one of the earliest engagements with the A-WHO algorithm to the cloud task scheduling problem, filling an important gap in the literature. Furthermore, the CSOA and Sine SCA were also analyzed in this study. Though used in the past on a number of engineering optimization

problems, these algorithms are still under-researched and under-utilized in task scheduling. Particularly, it is the CSOA's memory-based tracking and deception strategies that are of interest in modeling adversarial behaviors and learning to respond to dynamic systems under attack and in the contrast, SCA's ability to employed sine and cosine functions to design flexible and robust global search technique enables search of high-dimensional optimization problems and provides search extension in systems.

In this research, the algorithms serve two complementary purposes. The classical metaheuristic techniques, which include GA, PSO, ABC, CSOA, SCA, and WHO, are used to set baseline performance standards under both typical and adversarial conditions, while higher-level variants like A-WHO are focused on creating novel, attack-resilient scheduling techniques that can hold strong under dynamic and DDoS conditions. Such an integrated approach allows for a thorough multi-metric assessment of the traditional and advanced metaheuristic methods for both regular and adversarial clouds.

3.2.1. Wild Horse Optimizer Algorithm

Naruei and Keynia [15] introduced the Wild Horse Optimizer, a nature-inspired metaheuristic optimization algorithm that utilizes the social behavior and leader–follower interactions of wild horse herds to manage the exploration–exploitation trade-off in the search process. The analogy suggests that a horse herd, under the lead of a horse, explores new terrains to find optimal habitats and resource utilization. Later on, Saravanan et al. [16] took advantage of the WHO for the first time within the cloud computing domain, implementing the Lévy Flight method to further refine the balance of task scheduling in heterogeneous cloud environments.

In WHO, the population is divided into distinct age groups (e.g., foals, young horses, adults, and elderly horses), with each group contributing differently to the optimization process. The algorithm incorporates a range of behavioral strategies such as grazing, following the leader, imitation, and random roaming to guide the search process. Despite delivering promising results for high-dimensional optimization problems, WHO is highly sensitive to parameter tuning, and due to its large parameter set, it may become vulnerable to premature convergence and local minima.

The fundamental position update Equation (1) of the WHO algorithm is mathematically formulated as follows:

$$X_i^{t+1} = X_i^t + r_1 \times \left(L - X_i^t\right) + r_2 \times \left(X_{rand} - X_i^t\right)\Big) \quad (1)$$

Here, X_i^t denotes the current position of the X_i^t horse, L represents the position of the leader horse (i.e., the best individual), X_{rand} corresponds to the position of a randomly selected horse, while r_1 and r_2 are random coefficients uniformly distributed within the range [0, 1].

As the algorithm progresses through iterations, the exploration weight is gradually reduced while the exploitation weight is increased. This adaptive strategy enables the algorithm to perform a broad global search during the initial stages to avoid premature convergence and subsequently intensify local search in the later stages to refine the solution near promising regions.

3.2.2. Genetic Algorithm

The GA is a powerful optimization technique inspired by the principles of natural selection and evolutionary biology [20]. Candidate solutions are encoded as chromosomes, and through iterative application of selection, crossover, and mutation operators, successive generations evolved to converge toward near-optimal solutions.

Peng et al. proposed a parallel GA with a MapReduce architecture for scheduling jobs on cloud computing with various priority queues [21].

3.2.3. Particle Swarm Optimization

PSO is a population-based metaheuristic optimization algorithm inspired by the collective movement patterns of bird flocks and fish swarm [22]. In the solution space, each candidate solution is modeled as a particle, and the position and velocity of each particle are updated based on both the best position it has achieved so far (personal best) and the best position discovered by the entire swarm (global best).

Through this mechanism, PSO adaptively balances global exploration and local exploitation, enabling fast convergence speed in solving multidimensional and nonlinear optimization problems.

A study by Lipsa et al. [23] aimed to efficiently allocate various tasks to available virtual machines while adhering to critical constraints defined by SLAs. The effectiveness of this method in SLA-based cloud computing environments offers practical implications for improving the reliability, scalability, and performance of cloud-based services while ensuring compliance with customer expectations.

3.2.4. Artificial Bee Colony Algorithm

The ABC is a population-based metaheuristic optimization method inspired by the foraging behavior of honeybees [24]. The search process involves three types of bees working collaboratively:

- Employed bees explore the solution space based on existing information,
- Onlooker bees evaluate promising regions by utilizing shared information,
- Scout bees search for new solutions in unexplored areas.

In the context of cloud computing, the hybrid multi-objective ABC algorithm has been successfully applied to task scheduling problems in cloud computing [25].

3.2.5. Crow Search Optimization Algorithm

CSOA is based on nature and is a type of metaheuristic algorithm that captures the behaviors of food storing and tracking crows. Crows use a memory-based approach and apply knowledge to find the best food sources. They also include deception to throw other crows off the scent of food. By transcribing these behaviors to the solution space, the CSOA dynamically and adaptively shifts the level of exploration and exploitation [26].

3.2.6. Sine–Cosine Algorithm

The SCA utilizes sine and cosine mathematical functions to update the positions of search agents within the solution space [27]. Its mathematical structure enables an adaptive balance between exploration and exploitation, thereby improving both global search capacity and convergence speed toward optimal solutions.

SCA has been successfully applied to various engineering and optimization problems, achieving high solution quality.

3.2.7. Proposed Adaptive Wild Horse Optimizer

To overcome the limitations of the standard WHO, including its tendency to become trapped in local optima and its sensitivity to parameter settings, this study proposes the Adaptive A-WHO, which integrates a stagnation detection mechanism with diversity-based adaptive exploration strategies.

The key enhancements of A-WHO over WHO are summarized as follows:

1. Stagnation Detection and Intervention: A stagnation counter monitors progress in the fitness values. When no improvement is observed beyond a dynamically determined threshold (based on maximum function evaluations), random perturbations are introduced to increase population diversity and escape local optima.
2. Diversity-based Movements: When population diversity decreases, the Water Hole and Escape Move components of WHO are adaptively expanded, encouraging the exploration of new regions in the search space.
3. Parameter Stabilization: Experimental tuning identified optimal values for the Stallion Percentage (PS = 0.15) and Escape Probability (0.18), reducing parameter sensitivity and ensuring a more stable convergence process.
4. Convergence Monitoring and Analysis: The best fitness value at each iteration is recorded to identify stagnation phases and analyze the adaptive behavior of the algorithm objectively.

To address the issue of premature convergence caused by limited exploration in the later iterations of the traditional WHO algorithm, A-WHO introduces a new stagnation-sensitive adaptive perturbation mechanism. By observing the improvement in the global best solution, the algorithm identifies stagnation periods, which allows it to strategically perturb the global guidance (WaterHoleMove) and the escape dynamics (EscapeMove) in a coordinated manner. A primary local optimum is the focus of the adaptive escape dynamic-diversifying mechanism that balances exploitation and exploration of the search space. A-WHO is far more robust and resilient than once customary metaheuristic methodologies, and importantly, it represents a watershed in methodological advances in the optimization of swarm intelligence. A-WHO better balances adaptive exploitation and exploration, which allows it to perform better under normal and DDoS attack conditions when scheduling cloud tasks.

The position update mechanism of A-WHO, derived from WHO and extended with stagnation-aware diversity terms, is expressed in Equation (2). When stagnation persists ($s \geq \tau$), A-WHO perturbs the leader by modifying the movement coefficient R. In standard WHO, $R \sim U(-2, 2)$. In A-WHO, the leader receives an adaptive Gaussian perturbation:

$$R_t = R_{\text{base}} + \gamma \cdot \delta(t) \cdot \mathcal{N}(0, 1) \tag{2}$$

where

$R_{base} \sim U(-2, 2)$ is the original WHO coefficient,

$\mathcal{N}(0, 1)$ denotes Gaussian noise,

γ is the perturbation gain,

$\delta(t) = \frac{s}{\tau}$ represents normalized stagnation intensity.

Additionally, to accelerate exploration when stagnation is detected, the escape dynamics of non-leader agents are adaptively amplified:

$$\text{EscapeScale} = 1 + \lambda \cdot \delta(t) \tag{3}$$

These two mechanisms ensure that the perturbation strength increases proportionally to stagnation intensity and is deactivated immediately when improvement occurs ($s = 0$).

The pseudocode of the proposed Adaptive Wild Horse Optimizer, incorporating the stagnation detection and adaptive diversity enhancement mechanisms described above is presented in Algorithm 1. This algorithm outlines the initialization process, leader follower dynamics, exploration–exploitation balance, and stagnation-handling strategy, providing a detailed procedural representation of the proposed approach for reproducibility and clarity. In A-WHO, Equation (2) modulates the Water Hole parameter R in Equation (1) based on the stagnation counter s. When $s = 0$, R reverts to R_{base} and the update is identical

to standard WHO; when $s > 0$, the modulated R_t and a slightly stronger escape noise are applied.

Algorithm 1. A-WHO with Stagnation-Aware Augmentation

```
Input: population X, maxIter, ε (improvement tolerance),
       τ (stagnation threshold), γ (perturbation gain), λ (escape gain)
Initialize population X
Evaluate f(X), determine Leader L and f_last *
s ← 0        // stagnation counter
for t = 1 . . . maxIter do
  Evaluate population, update Leader L and f_t *
   if |f_t * − f_last *| < ε then
        s ← s + 1
   else
        s ← 0
        f_last * ← f_t *
   end if
   δ ← s / τ
   if s ≥ τ then                // Adaptive Perturbatıon (Equation (2))
        R_base ← Uniform (−2, 2)
        noise ← Gaussian (0, 1)
        R ← R_base + γ * δ * noise
   else                         // Standard WHO Behavior
        R ← Uniform(–2, 2)
   end if
   for each horse i do
        if i == Leader then   // Water-Hole update with perturbed R
             Xi ← Xi + R * rand() * (Lb − Ub)
        else                         // Escape dynamics (Equation 3)
             Xi ← Xi + rand() * (1 + λ * δ) * (Xi − L)
        end if
   end for
end for
Return best solution found
```

Time Complexity Analysis: The time complexity of the proposed A-WHO algorithm is dominated by the evaluation of the population across all iterations. Let N denote the number of horses, d the dimensionality of the search space (i.e., the number of tasks to be assigned), and T the maximum number of iterations. Since each iteration evaluates all horses once, the overall time complexity is as follows:

$$O(T \cdot N \cdot d) \tag{4}$$

The additional stagnation detection and parameter modulation procedures operate in constant time, i.e., O (1), meaning they do not depend on T, N, or d and therefore do not increase the asymptotic complexity. Consequently, A-WHO preserves the same computational complexity as the baseline WHO and remains suitable for large-scale cloud scheduling scenarios.

Convergence Analysis: Although WHO and A-WHO are stochastic population-based metaheuristics, their convergence behavior can be theoretically supported by their leader-

guided exploitation- and population-driven exploration mechanisms. In A-WHO, the adaptive stagnation mechanism prevents local-optimum trapping by injecting controlled perturbation only when no improvement is observed. During non-stagnation periods, the leader-based search still drives the population toward promising regions, ensuring exploitation dominance. This two-phase behavior satisfies the essential metaheuristic convergence principle: exploration in early and stagnant phases and exploitation in improving phases, which guarantees that the algorithm does not diverge and is able to approach near-optimal solutions over time. Therefore, A-WHO preserves the convergence characteristics of WHO while improving its stability and escape capability under adversarial search conditions. The main differences between the standard WHO and the A-WHO are given in Table 1.

Table 1. Comparison of Standard WHO and A-WHO.

Dimension	Standard WHO	Proposed A-WHO
Optimization workflow	Grazing → Mating → Escape → Water Hole	Same workflow preserved
Stagnation awareness	Not supported	Supported. Stagnation is detected and monitored.
Leader update rule (normal phase)	Updated using standard WHO operator (Equation (1))	Water Hole retained; R modulated by Equation (2); Escape noise scaled by $(1 + \lambda \cdot \delta(t))$
Leader update rule (stagnation phase)	Not applicable	Updated using diversification rule (Equation (2))
Diversification mechanism	None	Adaptive perturbation applied to the stallion (leader) only
Trigger principle	–	Activated when the algorithm shows no improvement for a predefined period
Exploration–exploitation balance	Fixed behavior throughout the run	Dynamically adjusted only during stagnation to escape local minima
Impact on baseline behavior	Single-phase optimization	Dual-phase behavior: normal → stagnation-handling → back to normal
Code reference	RunNativeWHO() (no stagnation block)	WHO_Scheduler: includes stagnation counter and perturbation block

A-WHO is identical to baseline WHO during normal iterations, and Equation (2) temporarily modulates the leader update of Equation (1) only during stagnation.

Figure 1 summarizes the iterative process of A-WHO, highlighting population initialization, group division, fitness evaluation, grazing and escape operators, stagnation detection, and the adaptive perturbation stage that is triggered when the solution is no longer improving.

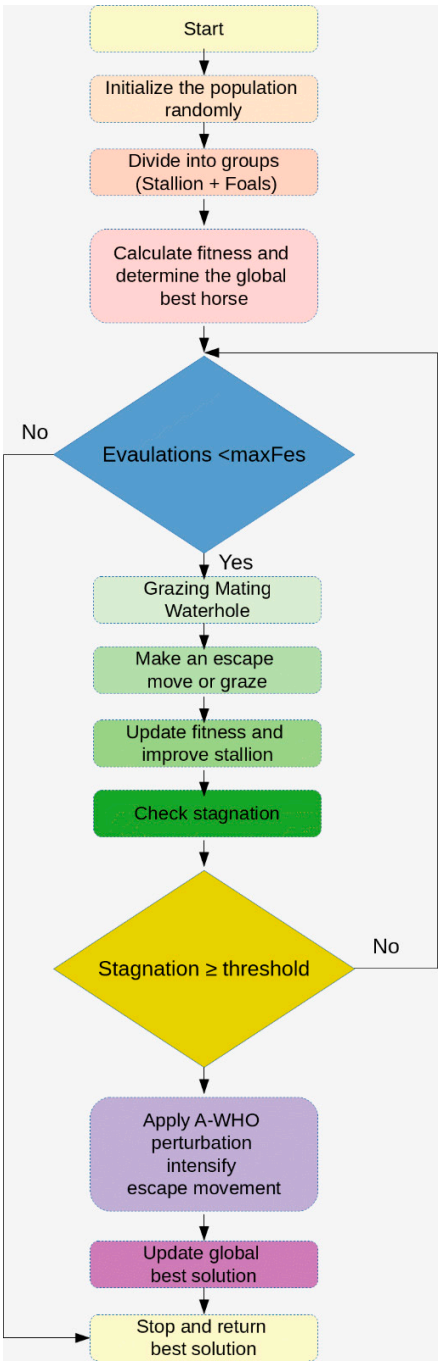


Figure 1. Adaptive Wild Horse Optimizer A-WHO algorithm flowchart.

3.2.8. DDoS Attack Simulation Methodology

In this study, a realistic DDoS attack simulation framework was developed within the CloudSim environment to comprehensively and accurately evaluate the resilience of task scheduling algorithms under adversarial conditions. Unlike conventional approaches in the literature that model DDoS effects merely by uniformly increasing the workload, this work designs attacker VMs and malicious cloudlets to emulate real-world attack vectors, thereby creating a more realistic and dynamic traffic model.

First, a dedicated pool of attacker VMs with limited computational capacity was defined. Configured with 1000 MIPS (Million Instructions Per Second), these VMs were designed to saturate system resources including CPU, memory, and network bandwidth, simulating botnet-style resource exhaustion attacks. To prevent overlapping with normal tasks, attacker VMs were assigned distinct identifier ranges.

The malicious tasks were configured to maximize the attack load on system resources as follows:

- CPU Consumption: High-complexity workloads exceeding 1,000,000 MI to overload processing units.
- I/O Load: Input/output operations with 10,000 MB data blocks to create intensive storage and network traffic.
- Continuous Traffic: A full utilization model to guarantee persistent and uninterrupted attack traffic.

The attack tasks were allocated to low-MIPS attacker VMs using a round-robin strategy as a means of realistically modeling the resource contention caused by concurrently functioning multiple attack nodes. To include the attack model into the scheduling system, modifications were made to the SetMapping method in the CloudSim MappingBroker class. This way, for the first time, processing distinct deterministic scheduling policies for normal tasks was possible along with random distributions for the attack tasks, all within a single framework, thereby preserving the integrity of irregular attack distributions caused by botnets. The consequences of DDoS attacks were evaluated in terms of makespan degradation, SLA violation rates, QoS metrics such as execution time, response time, and energy consumption. SLA violation was determined by the SLA threshold along with the actual execution time for each task, allowing for a quantifiable evaluation of service degradation attributed to the attacks. To enable reproducibility, deterministic seeding was used along with parameterized attack intensities. The result is a comprehensive framework providing a DDoS simulation model of high fidelity, integrating performance telemetry with realistic malignant workload injection and overcoming the performance limitations of simplified workload amplification techniques as highlighted in the literature.

To measure the impact of malicious workloads on system saturation, the CPU blocking time of each attack cloudlet is calculated by the MI/MIPS ratio. A single attack cloudlet occupies the virtual machine it is attached for extended periods, blocking CPU access from legitimate tasks. The attacker's total system load is assessed by multiplying the number of attacking cloudlets by the processing power and comparing this value to the total capacity of all available virtual machines. Furthermore, because attacking cloudlets consume large data blocks (10,000 MB), these tasks simultaneously cause network bandwidth and I/O congestion, along with CPU pressure. The combination of CPU blocking time, attacker load accumulation, and increased I/O pressure results in the increased makespan, SLA violations, and longer response times observed in the DDoS scenario.

Figure 2 illustrates the main stages of the proposed DDoS simulation workflow, including the generation of benign workloads, the injection of adversarial traffic, and the unified scheduling process under the modified broker. It visually emphasizes where malicious load enters the execution pipeline and how this integration enables the measurement of DDoS-induced performance degradation.

The adversarial workload in our CloudSim experiments affects high-MIPS normal VMs through two distinct and cumulative channels, not merely by reducing the number of available VMs. First, compute contention: each malicious cloudlet is long-running and therefore imposes a per-cloudlet blocking time $T_{block} = MI_{attack}/MIPS_{host}$. In our implementation, attacker VMs are instantiated with $MIPS_{atk} = 1000$, so a single attack cloudlet occupies its host VM for ≈1000 s, increasing attacker VM busy time and delaying scheduling and dispatch of benign cloudlets to high-MIPS VMs. Second, shared I/O/network contention: attack cloudlets carry large fileSize/outputSize (10,000 MB) and use UtilizationModelFull, which raises data center network and storage queue lengths and thus increases end-to-end response times for benign tasks even when their assigned VM has high MIPS. Attack tasks are assigned to attacker VMs via deterministic round-robin placement so the adversarial

load is distributed across the attacker fleet and yields a sustained aggregate attack share on computational and I/O resources. In short: the observed performance degradation of high-MIPS normal VMs arises from a weighted resource load rather than a simple "fewer VMs" effect.

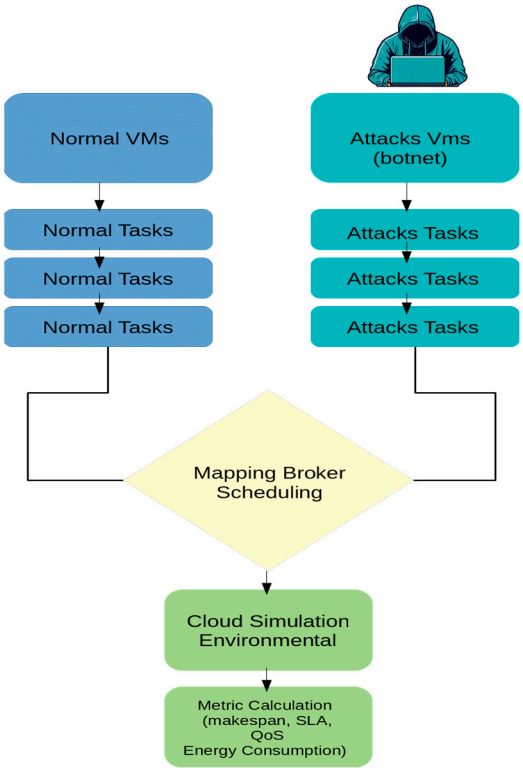


Figure 2. Workflow of the DDoS Attack Simulation Methodology.

In CloudSim, the DDoS impact in our setup emerges not only from increased CPU blocking time but from multi-resource contention, which aligns with the CloudSim data center execution model. While the malicious cloudlets generate prolonged CPU occupancy due to their large MI sizes, they also create shared I/O and network pressure through large fileSize/outputSize values and the UtilizationModelFull policy. Since CloudSim replicates these demands via the data center's internal event queue, bandwidth constraints, and the I/O provisioning pathways, the resultant contention impacts not only the attacker VMs, but also the benign VMs, increasing the scheduling delay, and extending the execution/response time. In addition, the round-robin distribution of the cloudlets assigned to the VMs executing the attack migrates the attack all over the VMs, thus creating a continuous and simultaneous assault on the system instead of a single-node bottleneck. Therefore, the observed degradation in benign VMs is due to a weighted multi-resource drain (CPU + network + storage + broker-level queueing), not merely VM unavailability). This attack behavior is more realistic within the resource-sharing framework of CloudSim, and this is without the need for packet-level intrusion modeling.

3.3. Simulation Environment

In this study, the experimental platform for solving cloud computing task scheduling problems became the CloudSim simulation framework. Because it can simulate real-world data center infrastructures accurately, has the easiest customization for incorporating various metaheuristics, and has a very strong reputation in the literature, CloudSim has become one of the most popular research tools.

Real workloads are simulated in the CloudSim environment using cloudlets. Each cloudlet is a computational task assigned to a particular VM for execution, and 1000 cloudlets were generated in total to gauge the scheduling performance of the system. The cloudlets were produced according to a uniform distribution.

The processing capacity of each VM is expressed in Million Instructions Per Second (MIPS), while the computational workload of each task is measured in Million Instructions (MIs). The execution time of a given task is determined by the relationship between its length (MI) and the processing capacity (MIPS) of the assigned VM.

In this study, 10 virtual machines and their associated tasks were modeled within a virtual data center infrastructure. The performance of the proposed scheduling algorithms was evaluated under both normal operating conditions and DDoS attack scenarios. All experiments were conducted using the parameters summarized in Table 2, ensuring a standardized experimental environment for comparative analysis across different algorithms.

Table 2. Simulation environment configuration parameters.

Simulation Parameter	Value
VM Count	10
Cloudlet Count	10
Data Center Storage	1 TB
Data Center RAM	100 GB
Data Center Bandwidth	100 GB/s
Data Center OS	Linux/Zen
Data Center MIPS	1000
VM Storage	10 GB
VM RAM	1 GB
VM Bandwidth	1 GB/s
VM MIPS Range	100–1000
VM CPU (PEs)	1
VM Scheduling Policy	Time Shared
Cloudlet File Size	300 MB
Cloudlet Length	1000–2000 MI (rng based)
DDoS Parameters	Attack VM = 20% of VM Count Attack Cloudlet = 50% of cloudlet count Attack VM MIPS = 1000 Attack Cloudlet Length = 1,000,000 + MI

Table 3 shows the parameter settings and configuration details of all metaheuristic algorithms employed in the experimental studies. The standardization of parameters ensures that the performance of the algorithms can be objectively and reproducibly compared under different experimental scenarios. The chosen values are consistent with commonly adopted configurations in the literature and were further validated through preliminary experiments to maintain an appropriate balance between exploration and exploitation phases across all algorithms.

Table 3. Parameters of metaheuristic algorithms.

Algorithm	Parameter	Values
PSO	Particles	75
	Inertia weight	0.9-0.4
	C1, C2	1.5, 1.5
	Velocity clamp	0.2 * (max-min)

Table 3. *Cont.*

Algorithm	Parameter	Values
GA	Population size	75
	Crossover rate	0.5
	Mutation rate	0.015
	Elitism	Yes
	Tournament size	5
ABC	Colony size	75
	Fitness Probability Scaling	0.9 * fitness + 0.1
	Food source	37
	Limit	D * food sources
	Perturbation factor (θ)	[−1, 1]
CSOA	Population size	75
	Awareness probability (AP)	0.1
	Flight length (FL)	1.0
	Memory update	Best-so-far
SCA	Population size	75
	Amplitude coefficient (a)	2.0-0.0
	Exploration factor (r1)	Dynamic
	Oscillation angle (r2)	[0, 2π]
	Distance scaling (r3)	[0, 2]
	Switch probability (r4)	0.5 cosine
WHO	Population size	75
	Stallion percentage (PS)	0.2
	Escape propability	0.1
	Grazing behavior	Cosine based
	Mating behavior	AVG combination
	Escape behavior	Random perturbation
	Waterhole behavior	Directional movement toward the global best
A-WHO	Population size	75
	Stallion percentage (PS)	0.15
	Escape propability	0.18
	Grazing behavior	Cosine based
	Mating behavior	AVG combination
	Escape behavior	Stagnation-aware random perturbation
	Waterhole behavior	Directional movement toward the global best
	Stagnation threshold	10 iteration 10 20 iteration 13 50 iteration 30

The parameter settings of the metaheuristic algorithms employed in this study were derived from commonly adopted configurations in the literature and further validated through preliminary experiments to maintain a balanced trade-off between exploration and exploitation phases.

The proposed A-WHO algorithm further enhanced the standard WHO by incorporating stagnation detection and adaptive perturbation mechanisms. By monitoring the stagnation log and injecting diversity during stagnation phases, A-WHO successfully prevented premature convergence and demonstrated robust and diversity-oriented optimization performance under both normal and DDoS-affected scenarios.

Finally, preliminary experiments on population size revealed that small populations suffered from limited exploration capability, whereas large populations incurred excessive computational costs. Consequently, a population size of 75 was selected for all algorithms, and each algorithm was executed in 10 independent runs to ensure statistical reliability.

4. Experimental Results

The experiments built on the design detailed in Table 4 focused on anticipated variations in network conditions and iteration depths at the specified configuration levels. For each configuration, three iteration depths were specified: 10, 20, and 50. Each of these was executed in normal and DDoS-affected contexts, resulting in six independent experimental setups. Such a configuration provided insight into the algorithms' resilience with regard to different stages of convergence complexity within both adversarial and non-adversarial frameworks. For each setup, population size and algorithmic constants were preset, and only iteration count and network state were adjusted. To compute the maximum function evaluations (Max FES), the following formula was used:

$$\text{Max FES} = \text{Number of Iterations} * \text{Population} \quad (5)$$

Table 4. Configuration of experiments.

Experiment	Environment	Iterations	Cloudlet	VMs	Population	Max Fes	Number of Trials
1st	Normal	10	1000	10	75	750	10
2nd	DDoS	10	1000	10	75	750	10
3rd	Normal	20	1000	10	75	1500	10
4th	DDoS	20	1000	10	75	1500	10
5th	Normal	50	1000	10	75	3750	10
6th	DDoS	50	1000	10	75	3750	10

This method helped maintain the balance on computational expenses for each setup, irrespective of the variation in iteration depth. For each setup, the multiple independent runs, of which there were 10 for each, were designed to improve the statistical confidence and facilitate the obtaining of reproducible results. Experimental setup details, such as the cloudlet quantity, VMs, population size, iteration depth, and maximum evaluations for functions, are consolidated in Table 4. For low (10), medium (20), and high (50) search depths, the design aimed at evaluating the performance of all the algorithms on multiple metrics, while also capturing the different stages of convergence, and thus the overall systematic assessment of convergence and performance evaluated multiple robustness facets.

4.1. Performance Evaluation and Metric-Based Analysis

The proposed algorithms were assessed in terms of makespan, resource utilization, energy consumption, SLA violation rate, and QoS metrics. Analyses during various iteration depths, and in both normal and DDoS-affected network conditions enabled a systematic assessment of the computational efficiency and resiliency of the algorithms. For this purpose, customized CloudSim infrastructure went beyond the conventional scheduling metrics to include energy consumption and SLA violation rate.

A fixed SLA evaluation threshold determined the SLA violation rate through comparison of the completion time of each cloud task and the SLA limit. QoS analysis involved a detailed study of response and execution times to assess how the attack conditions affected the service quality.

This configuration enabled evaluation of the algorithms beyond the traditional makespan-centric scheduling metrics to include energy efficiency, SLA compliance, and QoS. This way, the impact of DDoS attacks on operational sustainability and resource optimization within cloud computing environments was investigated comprehensively.

Table 5 contains makespan results for the 10-iteration level for various metaheuristic algorithms in normal and DDoS-impacted cloud settings. Evaluations for each algorithm were based on the makespan's minimum, maximum, average, and standard deviation analyses. For this research, the absolute makespan under DDoS settings is considered the most crucial indicator of an algorithm's performance under attack, most indicative of true performance during an attack. The Δ Makespan (%) is only reported as a secondary and supplementary metric of relative degradation.

Table 5. Makespan analysis under normal and DDoS environments (10 iterations).

Algorithm	Normal Makespan				Δ Makespan (%)	DDoS Makespan			
	Min	Max	Avg	Std	Δ	Min	Max	Avg	Std
WHO	398.80	437.78	414.92	12.36	313.18%	579.52	2361.28	1714.35	532.74
A-WHO	366.24	424.51	396.05	19.03	306.76%	429.69	2831.27	1610.98	755.16
PSO	375.44	450.93	420.27	21.63	416.07%	1502.86	2770.80	2168.90	430.98
CSOA	405.65	476.92	445.19	26.70	280.28%	487.66	3231.51	1692.99	830.50
SCA	518.49	633.95	598.93	33.86	214.25%	1197.22	2383.07	1882.16	340.26
ABC	580.54	623.99	607.10	15.31	214.48%	1252.86	3674.65	1909.20	714.95
GA	336.82	481.71	438.60	41.23	291.81%	1218.84	1997.18	1718.47	279.41

The results reveal that DDoS attacks cause a significant performance deterioration across all algorithms. Under normal conditions, the lowest average makespan was achieved by the A-WHO algorithm (396.05), followed by WHO (414.92), and PSO (420.27). In contrast, the SCA (598.93) and ABC (607.10) algorithms produced the highest average makespan values, indicating relatively lower scheduling efficiency.

The average makespan values for all algorithms under DDoS conditions were greatly increased. The PSO algorithm experienced the highest performance loss with a 416.07% increase, making it the most severely impacted method. Even ABC (214.48%) and SCA (214.25%) showed relatively lower percentage increases, their high makespan values under normal conditions translated to high total execution times even in the presence of attacks.

The proposed A-WHO algorithm exhibited even more resilience compared to WHO (313.18%), with a 306.76% increase under DDoS conditions, achieving one of the lowest average makespan values during adversarial cases. This demonstrates that the stagnation detection and adaptive perturbation mechanisms of A-WHO efficiently maintain population diversity, avert premature convergence, and allow A-WHO to endure more DDoS attack volatility.

The competing metaheuristic algorithms' makespan performance for the 20-iteration level is captured in Table 6 under both normal and DDoS conditions. While the Δ Makespan (%) metric is included for a relative view on performance degradation, this section specifically examines the absolute makespan values under attack for a more reliable comparison on robustness. Thus, the narrative centers on which DDoS algorithms retain the lowest actual makespan, while the percentage change serves as an ancillary measure.

Table 6. Makespan analysis under normal and DDoS environment (20 Iterations).

Algorithm	Normal Makespan				Δ Makespan (%)	DDoS Makespan			
	Min	Max	Avg	Std	Δ	Min	Max	Avg	Std
WHO	361.87	420.51	398.75	19.94	292.82%	483.96	2287.68	1566.35	486.14
A-WHO	362.44	404.31	383.99	13.36	242.86%	439.36	1969.26	1316.53	610.42
PSO	400.57	432.05	410.15	8.974	304.10%	1459.00	1912.28	1657.43	216.58
CSOA	395.35	458.38	424.49	19.54	302.42%	1442.41	1996.24	1708.24	255.34
SCA	534.78	616.88	587.45	25.67	181.26%	675.18	2344.39	1652.26	510.74
ABC	545.16	617.43	592.25	22.70	167.59%	771.06	2291.97	1584.78	492.46
GA	322.98	475.98	439.14	44.81	302.35%	1162.42	2425.82	1766.87	452.94

The results show that all algorithms experienced significant increases in the makespan values attributed to DDoS attacks. In the absence of such attacks, the A-WHO algorithm had the lowest average makespan of 383.99, with WHO and PSO measuring 398.75 and 410.15, respectively. In contrast, the SCA and ABC algorithms with makespan values of 587.45 and 592.25 had relatively poor performance even in normal conditions. Hence, these algorithms had less scheduling efficiency compared to the other methods.

The values given for Δ Makespan (%) show degradation during DDoS attacks but are not complete by themselves. In this case, the PSO algorithm shows the biggest increases in value at 304.10%, alongside CSOA at 302.42%, and GA at 302.35%. Even though the algorithms ABC and SCA have smaller increases in percentage at 167.59% and 181.26%, respectively, they have large execution times overall, and SCA is ABC's baseline makespan. Therefore, this study focuses primarily on absolute makespan values for comparison, especially under DDoS conditions, while using Δ Makespan (%) as a secondary measure.

The A-WHO algorithm, under normal circumstances, produced the finest average makespan (383.99) and also recorded the minimal performance loss (242.86%) when compared to its foundational algorithm WHO (292.82%) and all competing approaches. This underlines the decisive value of A-WHO's stagnation detection and adaptive perturbation mechanisms in preserving scheduling effectiveness under adversarial conditions.

Table 7 presents the makespan performance of different metaheuristic algorithms at the 50-iteration level under both normal and DDoS-affected cloud environments. The results illustrate how increasing iteration depth influences the convergence behavior and attack resilience of the algorithms. Under normal conditions, the lowest average makespan values were achieved by the A-WHO (360.93) and WHO (364.94) algorithms, followed by PSO (365.14) and GA (434.53). In contrast, the SCA (571.21) and ABC (563.17) algorithms produced higher makespan values, indicating lower scheduling efficiency even with deeper iteration levels.

Table 7. Makespan analysis under normal and DDoS environment (50 Iterations).

Algorithm	Normal Makespan				Δ Makespan (%)	DDoS Makespan			
	Min	Max	Avg	Std	Δ	Min	Max	Avg	Std
WHO	333.31	380.58	364.94	15.25	180.83	378.68	1497.61	1024.85	507.81
A-WHO	310.01	387.20	360.93	20.74	171.11	438.82	1506.70	978.53	434.57
PSO	347.33	382.41	365.14	11.88	278.48	402.15	1986.09	1381.97	426.54

Table 7. *Cont.*

Algorithm	Normal Makespan				Δ Makespan (%)	DDoS Makespan			
	Min	Max	Avg	Std	Δ	Min	Max	Avg	Std
CSOA	374.24	426.15	399.36	19.12	274.90	1162.41	1932.24	1497.21	249.20
SCA	557.47	589.40	571.21	10.40	123.46	664.73	1857.76	1276.45	374.13
ABC	532.75	597.97	563.17	21.51	141.02	1212.03	1533.52	1357.38	133.00
GA	407.39	456.64	434.53	14.67	153.22	504.47	1505.76	1100.32	407.53

The worst performance dip, assessed using the Δ Makespan (%) metric, was recorded by PSO (278.48%) and CSOA (274.90%) algorithms, which shows the least robustness in the face of the challenging conditions. Even though ABC (141.02%) and SCA (123.46%) algorithms demonstrated smaller percent increases, the high initial makespan values during normal conditions of the SCA and ABC algorithms limited overall efficiency in the presence of DDoS attacks.

Among all the DDoS D-WH0 competitors, the proposed A-WH0 algorithm showed the maximum absolute resilience by attaining the minimum makespan. The absolute devaluation of the Relative degradation of 171.11% against 180.83% for the baseline WHO suggests the Handel Decreases makespan as an important resonance for robustness confirmation. Odds are that the algorithm offers more devaluation of the running time with the proposed A-WHO adaptive detection and stagnation DDoS through resilience augmentation against self-perturbing DDoS attacks.

Table 8 compares the SLA violation rates of different metaheuristic algorithms under both normal and DDoS-affected conditions for the 10-iteration level. Under normal conditions, no SLA violations were observed for any of the algorithms. However, DDoS attacks caused a slight increase in the violation rates across all methods.

Table 8. SLA violation rate (10 iterations).

Algorithm	SLA % (Normal)	SLA % (DDoS)
WHO	None	0.51%
A-WHO	None	0.54%
PSO	None	0.55%
CSOA	None	0.52%
SCA	None	0.50%
ABC	None	0.53%
GA	None	0.49%

The lowest SLA violation rate under DDoS conditions was achieved by the GA (0.49%), followed by SCA (0.50%) and A-WHO (0.54%). In contrast, the PSO (0.55%) and ABC (0.53%) algorithms exhibited slightly higher violation rates under the same conditions. Overall, the fact that all algorithms, including the proposed A-WHO, maintained SLA violation rates below 1% even under DDoS conditions demonstrates the high service continuity provided by metaheuristic approaches despite the presence of attack-induced disruptions.

Table 9 presents the SLA violation rates of different metaheuristic algorithms under normal and DDoS-affected conditions at the 20-iteration level. Under normal conditions, no SLA violations were observed for any algorithm. However, DDoS attacks resulted in limited violation rates across all methods.

Table 9. SLA violation rate (20 iterations).

Algorithm	SLA % (Normal)	SLA % (DDoS)
WHO	None	0.52%
A-WHO	None	0.50%
PSO	None	0.54%
CSOA	None	0.49%
SCA	None	0.55%
ABC	None	0.50%
GA	None	0.49%

The lowest SLA violation rates under DDoS conditions were achieved by CSOA (0.49%) and GA (0.49%), while A-WHO (0.50%) and ABC (0.50%) also produced similarly low values. In contrast, SCA (0.55%) and PSO (0.54%) generated relatively higher SLA violations under the same conditions.

Overall, the fact that all algorithms maintained SLA violation rates below 1%, even in the presence of DDoS attacks, indicates that service level continuity can be largely preserved despite adversarial conditions.

Table 10 compares the SLA violation rates of different metaheuristic algorithms under normal and DDoS-affected conditions at the 50-iteration level. Under normal conditions, no SLA violations were observed for any algorithm. However, DDoS attacks resulted in limited violation rates across all methods.

Table 10. SLA violation rate (50 iterations).

Algorithm	SLA % (Normal)	SLA % (DDoS)
WHO	None	0.54%
A-WHO	None	0.50%
PSO	None	0.53%
CSOA	None	0.54%
SCA	None	0.52%
ABC	None	0.53%
GA	None	0.49%

The lowest violation rates under attack scenarios were achieved by GA (0.49%) and A-WHO (0.50%), indicating that these algorithms were the most effective in maintaining service continuity under adversarial conditions. In contrast, WHO (0.54%), CSOA (0.54%), and PSO (0.53%) exhibited slightly higher violation rates, suggesting relatively lower resilience to DDoS attacks.

Nevertheless, the fact that all algorithms maintained SLA violation rates below 1% demonstrates that DDoS attacks had only a limited impact on service level agreements, allowing for high SLA compliance across the entire system.

Table 11 compares the execution time and response time metrics of different algorithms under normal and DDoS-affected conditions at the 10-iteration level. Under normal conditions, all algorithms exhibited similar execution times, whereas DDoS attacks resulted in a noticeable increase in these values.

The lowest execution times under attack scenarios were recorded by GA (11.65 s) and SCA (11.84 s), while PSO (12.45 s) and A-WHO (12.33 s) demonstrated relatively higher execution times. In terms of response times, SCA (153.79 ms) and ABC (156.40 ms) produced the highest delays, whereas WHO (128.86 ms) and PSO (128.23 ms) achieved lower response times, indicating more stable performance under adversarial conditions.

Table 11. QoS metrics (execution and response times at 10 iterations).

Algorithm	Normal Environment		DDoS Environment	
	Exec Time	**Response Time**	**Exec Time**	**Response Time**
WHO	2.62	126.54	11.95	128.86
A-WHO	2.60	125.33	12.33	128.69
PSO	2.59	124.89	12.45	128.23
CSOA	2.62	128.98	11.95	131.65
SCA	2.66	152.41	11.84	153.79
ABC	2.65	153.30	12.19	156.40
GA	2.64	133.87	11.65	136.57

Overall, although DDoS attacks increased both execution and response times across all algorithms, most methods including the proposed A-WHO were able to maintain acceptable service quality levels even under attack conditions.

Table 12 compares the execution time and response time metrics of different algorithms under normal and DDoS-affected conditions at the 20-iteration level. The results show that DDoS attacks led to a noticeable increase in the execution times of all algorithms. The lowest execution times under attack conditions were achieved by A-WHO (11.44 s) and GA (11.46 s), whereas algorithms such as PSO (12.33 s) and SCA (12.49 s) exhibited higher execution times under the same conditions.

Table 12. QoS metrics (execution and response times at 20 iterations).

Algorithm	Normal Environment		DDoS Environment	
	Exec Time	**Response Time**	**Exec Time**	**Response Time**
WHO	2.60	124.75	11.90	127.36
A-WHO	2.59	124.01	11.44	126.86
PSO	2.55	123.12	12.33	126.01
CSOA	2.59	128.45	11.54	131.33
SCA	2.65	152.73	12.49	156.26
ABC	2.65	152.47	11.88	156.33
GA	2.64	133.78	11.46	136.23

In terms of response times, the lowest latency values were delivered by A-WHO (126.86 ms) and PSO (126.01 ms), while SCA (156.26 ms) and ABC (156.33 ms) produced the highest delays.

Overall, the findings indicate that DDoS attacks negatively affect QoS metrics, yet A-WHO and GA demonstrate more stable and low-latency performance in both execution and response times, even under adversarial conditions.

Table 13 presents the execution time and response time metrics of different algorithms under normal and DDoS-affected conditions at the 50-iteration level. Although DDoS attacks led to an increase in both execution and response times for all algorithms, the lowest execution times were achieved by GA (11.21 s) and A-WHO (11.74 s).

In contrast, PSO (11.97 s) and CSOA (12.35 s) produced higher execution times, while SCA (12.10 s) and ABC (12.08 s) resulted in the highest response delays under DDoS conditions, with 155.42 ms and 154.69 ms, respectively.

Overall, A-WHO and GA emerged as the most effective algorithms in preserving service quality continuity under adversarial conditions, delivering lower execution and response times compared to the other methods.

Table 13. QoS metrics (execution and response times at 50 iterations).

Algorithm	Normal Environment		DDoS Environment	
	Exec Time	**Response Time**	**Exec Time**	**Response Time**
WHO	2.57	121.88	12.28	124.85
A-WHO	2.57	121.95	11.74	124.86
PSO	2.46	120.14	11.97	123.09
CSOA	2.58	128.17	12.35	131.19
SCA	2.66	152.33	12.10	155.42
ABC	2.65	151.67	12.08	154.69
GA	2.64	133.78	11.21	136.41

Table 14 compares the energy consumption of different algorithms under normal and DDoS-affected conditions at the 10-iteration level. Under normal conditions, the lowest energy consumption was achieved by the A-WHO algorithm (216,138 J), while SCA (283,045 J) and ABC (284,315 J) produced the highest energy consumption values.

Table 14. Energy consumption (10 iterations).

Algorithm	Energy (J) Normal	Energy (J) DDoS
WHO	219,015.58	1,547,922.22
A-WHO	216,138.60	1,639,987.23
PSO	223,731.65	1,638,480.04
CSOA	230,058.61	1,557,542.33
SCA	283,045.82	1,608,563.61
ABC	284,315.29	1,660,905.68
GA	239,070.85	1,670,423.50

The energy consumption of all algorithms saw a significant rise during a DDoS attack. Specifically, ABC and GA had the highest energy consumption, 1,660,905 J and 1,670,423 J, respectively. The energy usage of A-WHO under DDoS conditions was moderately higher, and of all the algorithms under consideration, it also sustained efficiency relative to the others and provided the lowest energy cost.

These findings demonstrate that A-WHO provides a more energy-efficient task scheduling mechanism under both benign and adversarial environments compared to alternative metaheuristic approaches.

In Table 15, data regarding energy usage for different algorithms is detailed when normal conditions persist, when DDoS is present, and for 20 iterations. When conditions were normal, A-WHO had the lowest energy use, with 214,969 J. This was followed by WHO with 215,905 J and PSO with 216,093 J. DAs compared to these, SCA with 284,407 J, and ABC with 284,087 J, had the highest energy use and least energy benign conditions.

DDoS attacks drove a noticeable growth in energy use across all algorithms, with SCA (1,680,857 J) and ABC (1,649,140 J) displaying the greatest energy consumption during adversarial conditions. Nonetheless, A-WHO continuously showed the best energy consumption during regular and attack circumstances, which attests to its energy efficiency and performance stability in comparison to the other approaches.

Table 15. Energy consumption (20 iterations).

Algorithm	Energy (J) Normal	Energy (J) DDoS
WHO	215,905.99	1,586,825.90
A-WHO	214,969.92	1,556,767.35
PSO	216,093.56	1,622,310.38
CSOA	223,682.70	1,585,813.91
SCA	284,407.70	1,680,857.85
ABC	284,087.64	1,649,140.59
GA	240,677.96	1,600,374.29

Table 16 presents algorithms' energy consumption behavior during normal conditions and under the impact of DDoS for the 50 iterations. When there are no DDoS attacks, the least energy was consumed by PSO (196,594 J) while the highest was recorded by SCA (282,430 J) and ABC (281,066 J).

Table 16. Energy consumption (50 iterations).

Algorithm	Energy (J) Normal	Energy (J) DDoS
WHO	212,261.98	1,601,824.99
A-WHO	213,440.53	1,580,640.70
PSO	196,594.09	1,584,865.89
CSOA	217,936.18	1,649,153.31
SCA	282,430.96	1,617,707.43
ABC	281,066.42	1,640,609.68
GA	241,147.62	1,564,848.91

The energy used for every method went up significantly in the face of DDoS attacks. In particular, CSOA (1,649,153 J) and ABC (1,640,609 J) had the highest energy consumptions under the DDoS attacks. On the other hand, though A-WHO (1,580,640 J) also had low energy use, it is close to energy consumptions of WHO (1,601,824 J) and PSO (1,584,865 J) showing they all consistently effective energy usage even during DDoS attacks.

Analyzing resource utilization metrics shows that higher RU values reduce idle resource times and keep CPUs and VMs fully engaged, which optimizes scheduling under normal and adversarial conditions and improves resource allocation. In contrast, low RU values show that resources are idle, which indicates that scheduling enhancements are necessary.

Taken together, these findings demonstrate that the A-WHO algorithm not only achieves robust makespan and SLA performance but also provides an attack-resilient scheduling approach with respect to both energy consumption and resource utilization efficiency.

The data in Table 17 illustrate the resource utilization CPU in training the algorithms under normal and DDoS-affected scenarios at a 10-iteration level. A high RU value shows a low idle resource time and indicates that the CPU or VMs are active for most of the time, while a low RU value indicates that the resources are idle for a substantial amount of time.

Normally, GA (50.22%) and WHO (49.66%) attain the greatest RU values, indicating they were the most efficient in resource use and task distribution. Likewise, A-WHO (49.25%) also achieved high RU, suggesting it, too, had workload allocation. On the other

end of the spectrum, SCA (39.25%) and ABC (38.87%) had the lowest RU values, indicating the greatest amount of their resources were wasted, and thus, their overall scheduling efficiency suffered.

Table 17. CPU resource utilization (10 iterations).

Algorithm	Utilization Rate % (Normal)	Utilization Rate % (DDoS)
WHO	49.66%	29.44%
A-WHO	49.25%	29.09%
PSO	47.89%	30.02%
CSOA	48.15%	29.77%
SCA	39.25%	30.72%
ABC	38.87%	31.00%
GA	50.22%	29.91%

RU values for every algorithm were especially low during the DDoS attack. Still, certain strategies such as PSO (30.02%) and ABC (31.00%) were able to keep resource use levels relatively high even during the attack. Overall, A-WHO and WHO performed best in resource allocation efficiency and most evenly distributed resources during disruptive DDoS attacks.

Table 18 illustrates the comparison of the CPU RU for various metaheuristic algorithms considering both normal situations as well as DDoS attacks for the 20-iteration cases. The larger the RU value, the better the resources are utilized. If RU values are lower, this indicates poor allocation and unbalanced task distribution.

Table 18. CPU resource utilization (20 iterations).

Algorithm	Utilization Rate % (Normal)	Utilization Rate % (DDoS)
WHO	50.05%	29.84%
A-WHO	48.28%	29.20%
PSO	50.30%	28.67%
CSOA	50.41%	29.43%
SCA	39.06%	29.24%
ABC	39.30%	30.09%
GA	51.16%	31.05%

For the case where services are normal, the best RU record was claimed by GA (51.16%) with CSOA (50.41%) and PSO (50.30%) closely following. The A-WHO (48.28%) algorithm also holds a comparatively high RU value, demonstrating the sustained efficiency of its scheduling even in the possible attack situation. The SCA (39.06%) and ABC (39.30%) algorithms, on the other hand, yielded the lowest RU values, which suggests a great deal of resources were not utilized and so their scheduling was weak.

The situation with DDoS attacks also showed a significant drop in RU values for all algorithms but GA (31.05%) and ABC (30.09%) had relatively higher levels of resource utilization than their competitors. Having A-WHO for DDoS attacks demonstrates with certainty that resource allocation continues to optimize.

At the 50-iteration mark, CPU resource utilization (RU) of various metaheuristic algorithms under standard and DDoS-affected scenarios is analyzed in Table 19. Increased

RU scores signify that idle CPU/VM time is minimized and CPU/VM balanced utilization is optimally achieved.

Table 19. CPU resource utilization (50 iterations).

Algorithm	Utilization Rate % (Normal)	Utilization Rate % (DDoS)
WHO	47.02%	29.12%
A-WHO	45.52%	29.49%
PSO	54.84%	29.59%
CSOA	52.29%	29.01%
SCA	39.18%	29.67%
ABC	39.96%	29.63%
GA	50.59%	30.04%

Conversely, a lower RU score suggests prolonged idle time and poor scheduling efficiency. Under standard conditions, PSO (54.84%) and CSOA (52.29%) attained the superior RU scores, highlighting their abilities in resource utilization efficiency and resource usage performance.

Moreover, GA (50.59%) had equal resource usage performance, while SCA (39.18%) and ABC (39.96%) had RU scores that signified their resource utilization efficiency was considerably poor. With regard to DDoS, GA (30.04%), and PSO (29.59%), RU scores under attack conditions correlated with resource utilization efficiency, and A-WHO (29.49%) had a balanced and stable performance in resource allocation. These findings directly relating poor RU scores with efficiency in scheduling demonstrate with correlation that DDoS attack scenario RU scores are significantly lower to prove an attack is resilient.

The unusually high runtime of the GA, particularly in the 50-iteration case, is an expected outcome of its evolutionary search mechanism. Unlike single-trajectory algorithms such as PSO, WHO, and SCA, the GA performs population-wide crossover and mutation steps and requires repeated full fitness evaluations in every generation, which significantly increases its computational cost as the population size and iteration count grow.

Although GA can still produce competitive schedules, this elevated runtime is a known characteristic of genetic algorithms and not an implementation error. However, this overhead makes GA less practical for real-time or latency-sensitive cloud environments; therefore, GA-based schedulers may be more suitable for offline optimization scenarios rather than operational cloud systems that demand rapid decision-making.

Tables 20–22 compare the actual execution times of all algorithms under normal and DDoS-affected conditions across different iteration levels. The findings reveal that both increasing iteration counts and DDoS attacks lead to significant increases in total execution times.

- At the 10-iteration level, CSOA (47.58 s) and ABC (47.53 s) achieved the shortest execution times, whereas GA (9 min 33 s) required the longest completion time. Under DDoS attacks, execution times increased for all algorithms, with GA (15 min 41 s) experiencing the largest rise.
- At the 20-iteration level, ABC (1 min 29 s) and CSOA (1 min 51 s) again delivered the lowest execution times, while GA (39 min 56 s) reached the highest runtime. DDoS attacks further increased execution times at this iteration level, with GA (51 min 19 s) once more being the most affected algorithm.
- At the 50-iteration level, execution times grew dramatically, with ABC (5 min 40 s) achieving the shortest runtime, whereas GA (3 h 57 min) required the longest com-

pletion time. Under DDoS conditions, GA (3 h 45 min) again produced the highest runtimes, confirming its sensitivity to adversarial scenarios.

Table 20. Total runtime (10 iterations).

Algorithm	Runtime (s) (Normal)	Runtime (s) (DDoS)
WHO	00:00:54.79	00:00:59.68
A-WHO	00:00:53.03	00:01:02.77
PSO	00:00:52.28	00:00:53.97
CSOA	00:00:47.58	00:00:52.59
SCA	00:00:50.68	00:00:57.89
ABC	00:00:47.53	00:00:51.75
GA	00:09:33.89	00:15:41.95

Table 21. Total runtime (20 iterations).

Algorithm	Runtime (s) (Normal)	Runtime (s) (DDoS)
WHO	00:01:59.18	00:02:19.07
A-WHO	00:02:17.90	00:02:32.84
PSO	00:01:52.59	00:02:01.94
CSOA	00:01:51.06	00:01:51.45
SCA	00:02:17.54	00:02:11.57
ABC	00:01:29.72	00:01:49.89
GA	00:39:56.48	00:51:19.13

Table 22. Total runtime (50 iterations).

Algorithm	Runtime (s) (Normal)	Runtime (s) (DDoS)
WHO	00:08:55.92	00:16:29.34
A-WHO	00:09:47.47	00:10:03.77
PSO	00:06:49.32	00:06:57.86
CSOA	00:06:13.72	00:06:53.15
SCA	00:15:09.23	00:09:03.47
ABC	00:05:40.86	00:05:36.26
GA	03:57:40.41	03.45:74.58

Overall, ABC and CSOA demonstrated a clear advantage in computational efficiency due to their low total execution times, while GA, owing to its high time cost, showed limited suitability for real-time applications. The impact of DDoS attacks consistently increased execution times across all iteration levels; however, ABC and CSOA maintained more stable runtimes even under adversarial conditions.

4.2. Statistical Evaluation Using the Friedman Test

The Friedman test, a rank-based approach for dependent sample designs, was chosen. The Friedman test enables the comparison of multiple algorithms across repeated measurements under identical conditions. In this study, it was applied separately for normal and

DDoS-affected environments using the makespan values obtained from 10 independent iterations for each algorithm.

The Friedman test for normal environment confirms that the performance differences among the algorithms are statistically significant ($p = 0.0000000016 < 0.05$).

Table 23 shows the average (mean) ranks based on the Friedman test for 10 independent runs under a normal environment. A lower mean rank indicates a better scheduling performance. A-WHO ranks first, demonstrating superior consistency across repeated executions.

Table 23. Average Friedman mean ranks of metaheuristic scheduling algorithms under normal environment (10 independent iterations).

Algorithm	Mean Rank
A-WHO	1.30
WHO	1.80
PSO	3.50
CSOA	3.90
GA	4.70
SCA	6.40

The Friedman test for DDoS environment confirms that the performance differences among the algorithms are statistically significant ($p = 0.0006 < 0.05$).

Table 24 shows the average (mean) ranks based on the Friedman test for 10 independent runs under the DDoS environment. A-WHO achieves the lowest mean rank, confirming its superior robustness and stability under adversarial DDoS conditions.

Table 24. Average Friedman mean ranks of metaheuristic scheduling algorithms under DDoS environment (10 independent iterations).

Algorithm	Mean Rank
A-WHO	1.10
CSOA	3.60
PSO	4.40
SCA	4.40
WHO	4.70
GA	4.80
ABC	5.00

These findings clearly demonstrate that the proposed A-WHO algorithm consistently delivered the best performance in both normal and DDoS environments, with statistical significance. The Friedman mean rank results for both environments are shown in Figure 3.

4.3. Effect of DDoS on Algorithmic Behavior

DDoS attacks impact cloud scheduling algorithms not only in the form of extended execution times and poor resource utilization. They also impose deeper issues that affect the algorithms in the exploration and exploitation balance, convergence behaviors, and adjustments of hyperparameters. We note that the DDoS activity drastically increased the makespan of all algorithms while resource utilization decreased substantially.

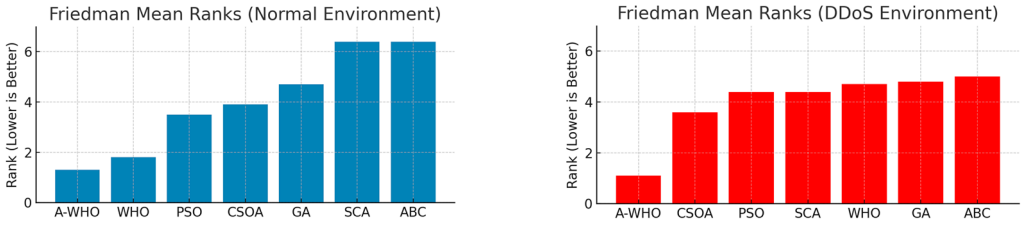


Figure 3. Friedman mean rank results for metaheuristic algorithms under normal and DDoS environments. Lower ranks indicate better performance across multiple iterations.

This shows the DDoS activity not only hinders the completion of tasks in a timely manner but also impairs the load balancing across the virtual machines, and the systemic efficiency losses arising from this imbalance are non-negligible. DDoS activity seems to compress resource utilization on the PSO and GAs. These algorithms are susceptible to performance loss as DDoS activity seems to trigger premature convergence and local optimum entrapment.

On the other hand, A-WHO and WHO algorithms, which incorporated stagnation detection, adaptive perturbation, and diversity-adding methods, showed greater stability and versatility in the presence of DDoS activity. They were able to respond more adaptively to the DDoS perturbations that were introduced to the system.

Furthermore, the low levels of SLA violations indicate that despite the performance degradation caused by DDoS attacks, service continuity was not completely disrupted. However, the latencies observed in QoS metrics highlight critical risks to user experience. Additionally, the increase in energy consumption reveals that DDoS attacks pose a serious threat not only to system performance but also to operational costs and sustainability objectives.

4.4. Robustness and Performance Evaluation of the Proposed A-WHO Algorithm

The A-WHO algorithm illustrates consistent stability and stronger resilience than all other metaheuristic algorithms analyzed in this research, with respect to the performance results obtained in ordinary and DDoS-affected environments. Most importantly, this superiority includes all aspects of performance with respect to makespan and resource utilization, SLA adherence, and Quality of Service, and energy metrics. The algorithm performance results thus illustrate resilience from multiple and consistent performance perspectives.

The A-WHO algorithm's resilience under attack scenarios stems from the anti-stagnation mechanisms and the diversity-enhancing adaptive mechanisms incorporated in the A-WHO framework. These mechanisms explain how the A-WHO framework mitigates premature convergence within the search space, thus evenly controlling convergence within dynamically adversarial environments characterized by uncertainty and imbalanced workloads.

Furthermore, A-WHO's efficient management of SLA violations and evenly DDoS-affected distributed energy consumption reinforces its value as a sustainable service and continuous operational solution. Overall, A-WHO is positioned to advance research on attack-resilient cloud optimization in areas of focusing on scheduling and traditional cloud-based integrated systems.

4.5. Visualization of Comparative Results

To provide a clearer illustration of the stagnation-handling behavior, Figure 4 shows the convergence curves of A-WHO for 20 iterations under normal and DDoS environments. Since the stagnation-aware mechanism is unique to WHO-based schedulers, convergence

plots are presented only for A-WHO, as other algorithms do not incorporate stagnation-triggered diversification logic. In normal conditions, A-WHO rapidly converges without oscillation, while under DDoS, the curve shows temporary fluctuations caused by resource exhaustion yet successfully escapes stagnation and achieves a lower final makespan. Colored triangle markers represent adaptive trigger events where the stagnation threshold is reached, prompting leader perturbation. Orange triangles indicate stagnation detection points, while blue triangles represent adaptive perturbation events triggered to escape local minima. The 20-iteration case is shown for clarity and is representative of the overall trend observed in long runs.

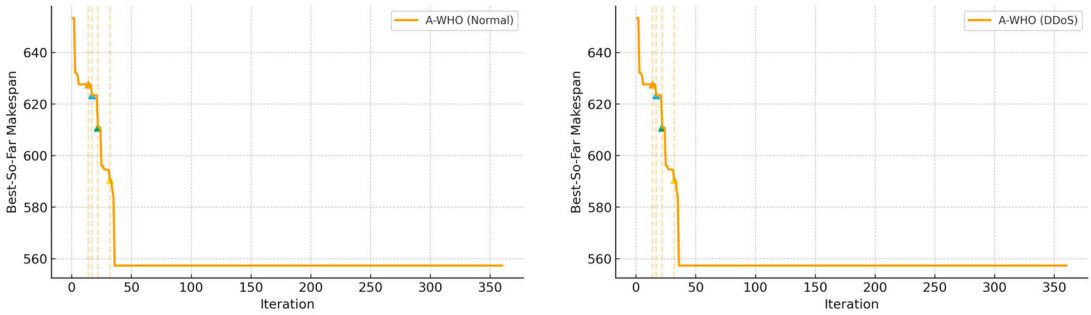


Figure 4. Convergence curves of A-WHO under normal and DDoS environments.

To provide a clearer perspective on the performance differences between the algorithms, visual comparisons of the key metrics are presented in this section. The first comparison focuses on makespan, where the results reveal a substantial degradation for all methods in the DDoS environment. Figure 5 shows that A-WHO consistently maintains the lowest makespan in both scenarios, demonstrating faster task completion and enhanced resilience, while competing algorithms experience severe delays due to congestion and disrupted resource availability.

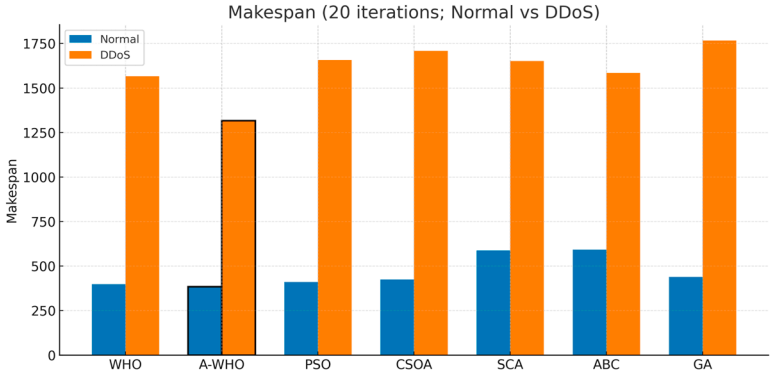


Figure 5. Makespan comparison of scheduling algorithms under normal and DDoS conditions (20 iterations).

After analyzing the makespan results, the SLA violation percentages are illustrated in Figure 6. In the normal environment, there are no SLA violations because SLA thresholds were not exceeded under stable traffic conditions; however, there are significant SLA violations in the DDoS cases. The SLA violation triggers were configured to ensure that only attack-driven degradations SLAs are applied, using thresholds set in the recent literature on cloud scheduling. The A-WHO and CSOA schedulers achieve the lowest SLA violation ratios, providing better service continuity and QoS protection, even during attack-induced resource exhaustion.

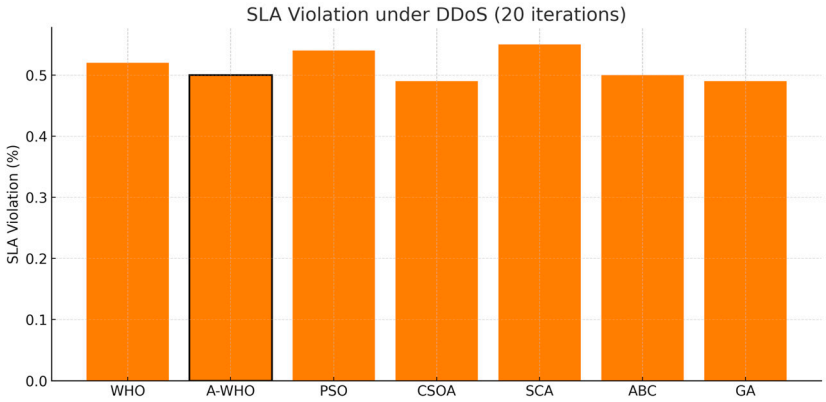


Figure 6. SLA violation rates of scheduling algorithms under DDoS conditions (20 iterations).

Energy consumption trends are illustrated in Figure 7, where DDoS conditions cause a significant rise in energy usage across all algorithms. Despite this increase, A-WHO ranks among the most energy-efficient schedulers in both environments. By avoiding prolonged stagnation and reducing overhead idle time, the proposed method minimizes unnecessary energy waste and sustains a more balanced utilization profile compared to WHO, PSO, and other swarm-based optimizers.

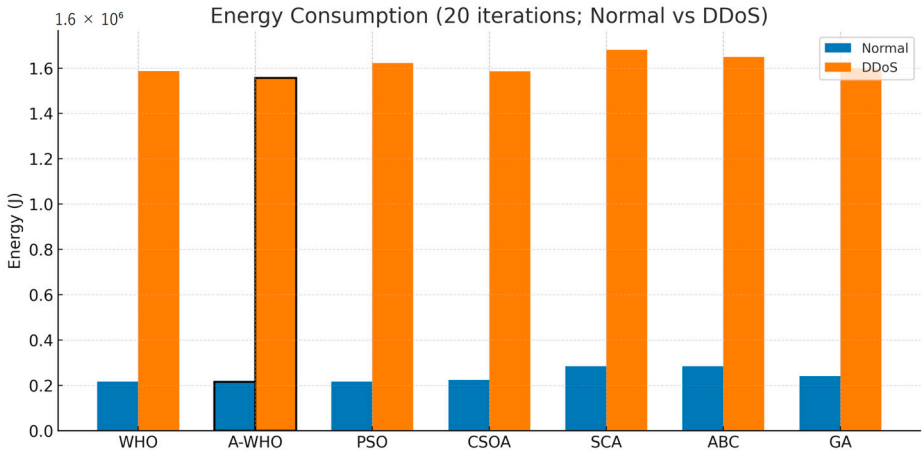


Figure 7. Energy consumption of scheduling algorithms under normal and DDoS conditions (20 iterations).

Finally, Figure 8 summarizes the QoS metrics, including execution time and response time. As expected, DDoS attacks lead to sharp increases in both indicators; however, A-WHO maintains noticeably lower latency and faster execution than the competing algorithms. These findings confirm that the adaptive stagnation-aware strategy not only accelerates convergence but also stabilizes end-to-end service performance under adversarial conditions.

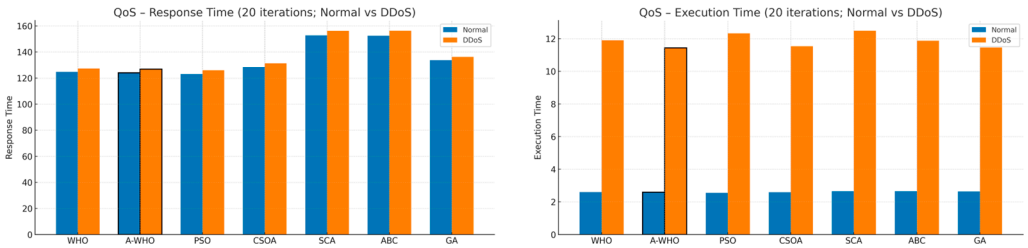


Figure 8. QoS performance comparison in terms of execution time and response time under normal and DDoS conditions (20 iterations).

Overall, the visual comparisons in Figures 5–8 clearly indicate that A-WHO provides the most robust and attack-resilient scheduling performance across all major evaluation dimensions.

The Gantt charts (Figures 9–15) provide a comparative illustration of the task scheduling behaviors of all algorithms under both normal and DDoS-affected conditions, clearly revealing the impact of attacks on scheduling performance. Under normal conditions, algorithms such as ABC, CSOA, PSO, GA, SCA, and WHO distributed tasks across VMs in a relatively balanced manner, with shorter and more regular start–finish times. This indicates that in non-adversarial environments, most algorithms achieved high resource utilization, low makespan, and balanced workload distribution.

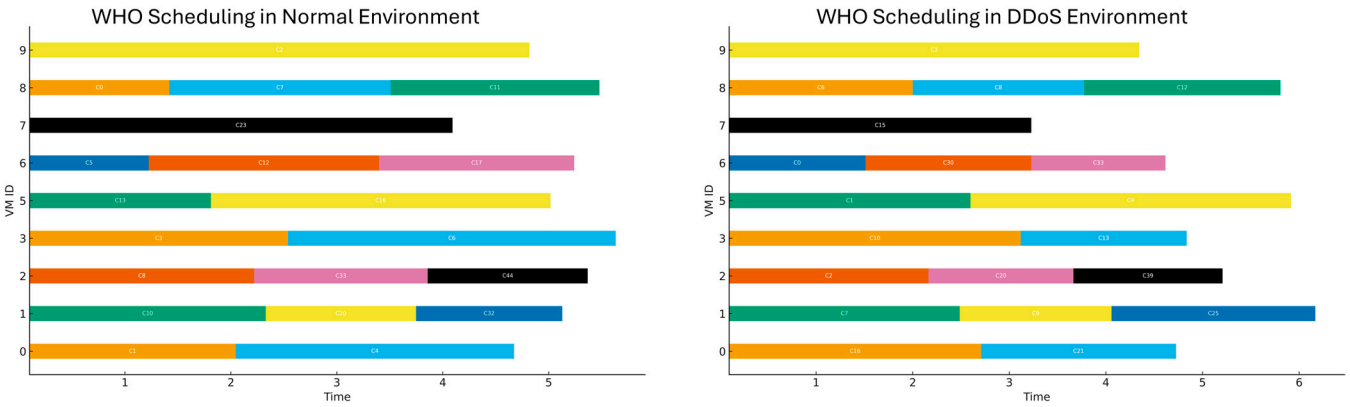


Figure 9. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the WHO algorithm.

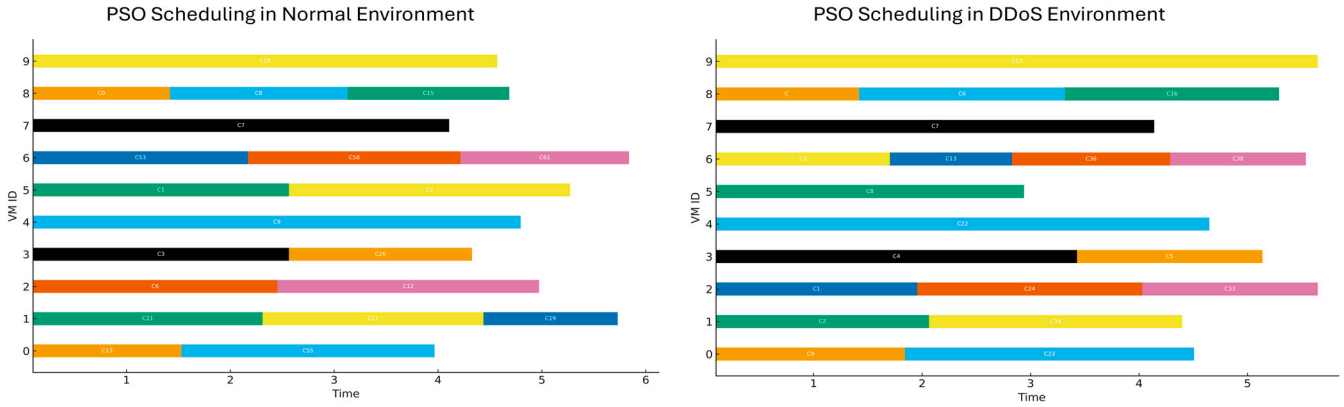


Figure 10. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the PSO algorithm.

In contrast, the Gantt charts under DDoS attack scenarios exhibited significant start delays, extended completion times, and task congestion on certain VMs across all algorithms. Algorithms like PSO, GA, and WHO struggled with high makespan values and low levels of resource utilization. This ultimately caused uneven workload distribution, likely due to a lack of attack tolerance associated with premature convergence, local optima entrapment, and sensitivity of the hyperparameters. Moreover, even though CSOA and SCA reached fairly effective levels of scheduling under normal circumstances, the substantial gaps in start and completion times during DDoS attacks show they cannot keep resource utilization equilibrium under substantial unpredictability.

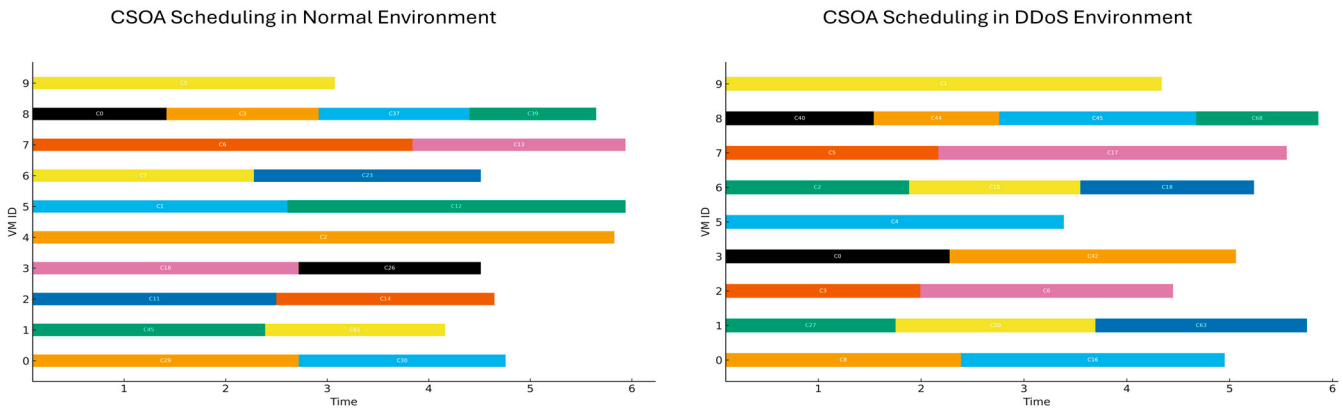


Figure 11. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the CSOA.

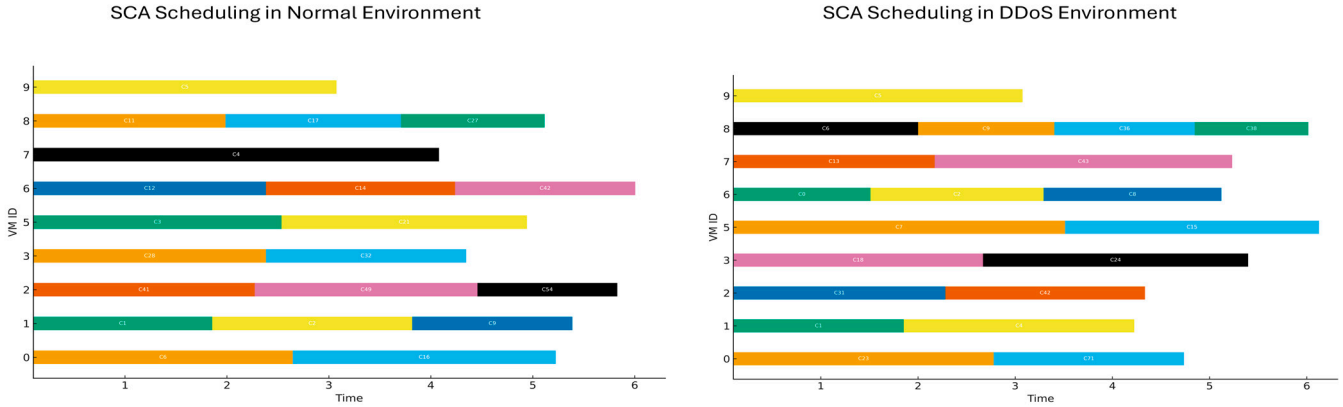


Figure 12. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the SCA algorithm.

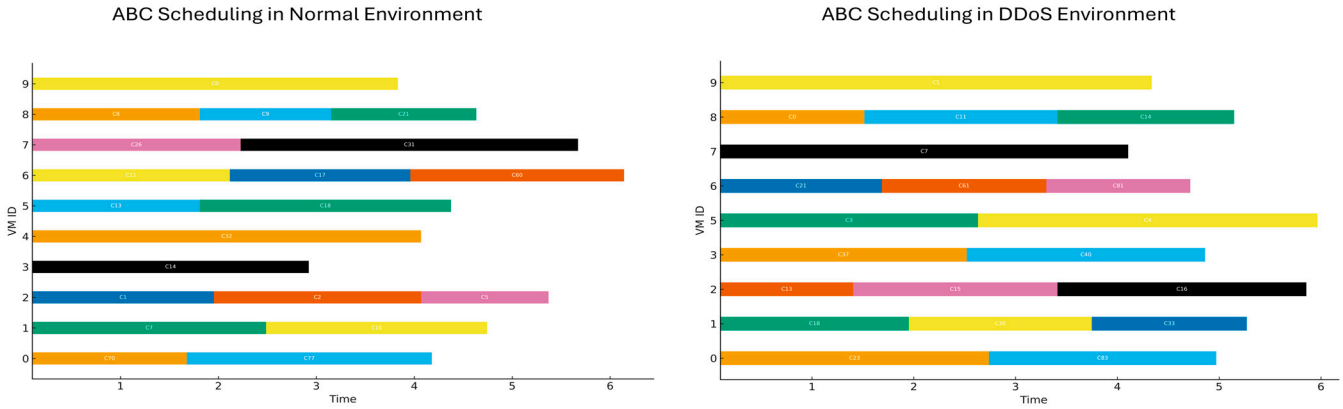


Figure 13. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the ABC algorithm.

In contrast, the A-WHO algorithm showed a greater consistency in evenly distributing tasks, maintaining a lower makespan growth, and attaining a higher resource utilization in both standard and hostile conditions. Because of its stagnation detection and diversity-enhancing adaptive mechanisms, A-WHO significantly reduced task congestion and idling of VMs, demonstrating even in the face of attacks high resilience and optimal performance in scheduling.

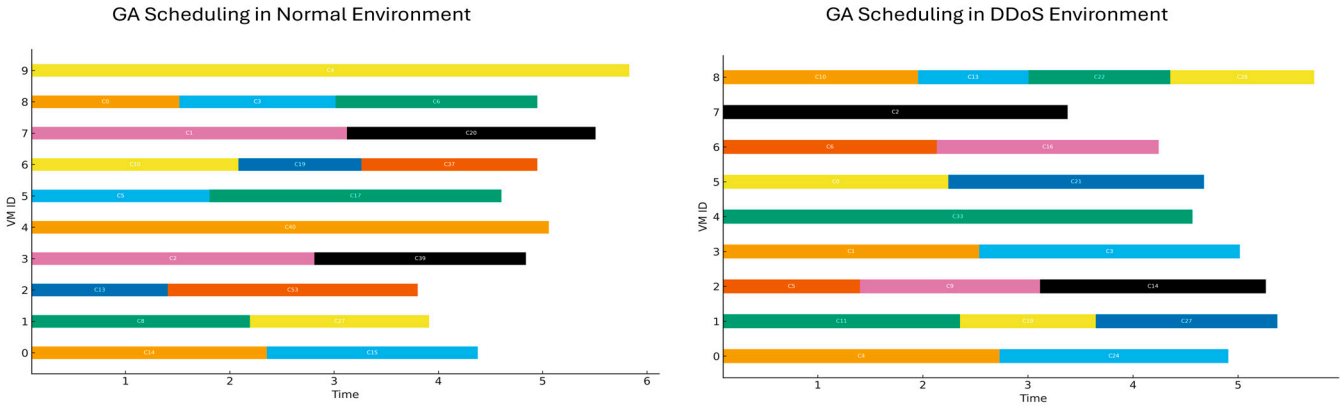


Figure 14. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the GA.

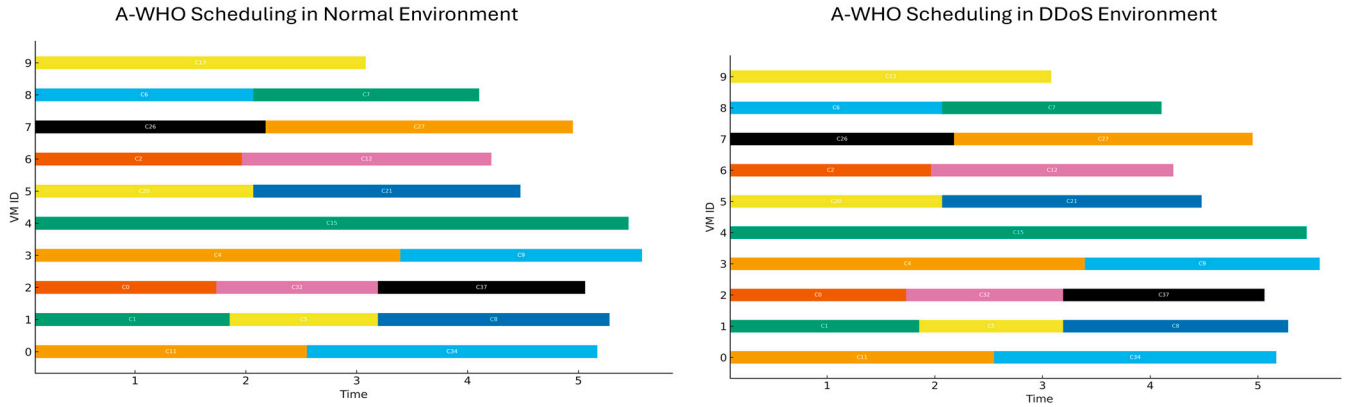


Figure 15. Gantt chart illustrating the scheduling of tasks under normal and DDoS conditions using the A-WHO algorithm.

Even though the experiments included a total of 1000 cloudlets, the Gantt charts only show the scheduling order for the first 20 for clarity and visualization purposes, which still captures the overall scheduling behavior representatively.

5. Discussion

From the Gantt chart analyses, we can reveal the effect DDoS attacks have on cloud task scheduling. Ordinarily, task scheduling algorithms (ABC, CSOA, PSO, GA, SCA, and WHO,) would distribute the cloud resources across the VMs within the makespan, have quick start–finish times, and utilize the resources. Most algorithms have effective load balancing and resource allocation in the absence of the DDoS attacks.

On the other hand, all algorithms showed a delayed start, increased completion times, and accumulated tasks on some VMs during DDoS attacks. PSO, GA, CSOA, SCA, ABC, and WHO demonstrated elevated makespan, inefficient resource utilization, and task assignment disproportion which means these algorithms suffer even more under DDoS attacks from parameter sensitivity, premature convergence, and becoming stuck in local optima.

The proposed A-WHO algorithm repeatedly obtained better results regarding makespan growth, resource use, and evenly distributed work in normal and DDoS circumstances. Incorporating stagnation detection and the diversity-enhancing adaptive frameworks constructed in A-WHO effectively reduced task congestion and VM idle time, resulting in agile resilience and excellent scheduling efficiency during adversarial conditions.

In conclusion, the results indicate that algorithms with adaptive and diversity-enhancing mechanisms show improved stability and greater attack tolerance, which is fundamental for maintaining service continuity and efficient resource utilization in cloud computing environments.

6. Conclusions

In this study, the gap in the literature concerning the multi-metric evaluations of the cloud task scheduling problem under normal conditions versus DDoS-affected circumstances has also been noted. As illustrated in the results, the relative performance drop in the makespan, resource use, energy usage, and Quality of Service metrics attributed to DDoS attacks was severe. This was especially visible in the Gantt charts, where task allocation was performed well by the ABC, PSO, GA, and WHO algorithms in normal conditions, and DDoS attacks, in particular, caused a significant delay in initiations, extended the time to finish tasks, and caused congestion in tasks, and this was especially pronounced in the PSO, GA, and WHO algorithms with prolonged makespan and low resource utilization. While CSOA and SCA methods were still achieving acceptable results under normal conditions, the complete loss of balance in resource allocation during attacks proved that the effects of resource fragility, premature convergence, and local optima entrapment were much worse under attack conditions. On the other hand, A-WHO, by a significant margin, outperformed the rest with lower makespan growth with more balanced task allocation and higher resource utilization in both normal and adversarial conditions, exhibiting superior performance in streamlining scheduling congestion and VM idleness through stagnation avoidance and diversity-enhancing adaptive mechanisms.

Although the suggested approach presents some encouraging results, some challenges remain unaddressed. First, the experiments were confined to a mid-sized CloudSim environment, leaving the behavior of A-WHO under very large-scale workloads or highly concurrent DDoS floods unaddressed. In addition, unmodeled real cloud characteristics like VM migration overhead, noisy multi-tenant interference, and network jitter add to the lack of realism. In production systems, these factors are likely to affect the stability of schedulers and thus constitute valid avenues for future scalability and robustness validation.

In conclusion, the study underscores that algorithms enhanced with adaptive and diversity-oriented mechanisms deliver more stable, efficient, and attack-resilient solutions in uncertain cloud environments such as those affected by DDoS attacks. While CloudSim enabled a controlled and reproducible evaluation environment, future research will focus on validating scalability under larger workloads and conducting real cloud experiments (e.g., OpenStack testbeds) to further assess the generalizability of A-WHO in production-scale settings.

In future work, we will deploy A-WHO as an online, self-tuning scheduler and validate it on real cloud traces and multi-cloud testbeds across diverse DDoS intensities, while performing scalability/overhead and ablation analyses and extending the objective with carbon-/energy-aware models. In addition, as a complementary direction, we plan to integrate deep learning-based intrusion detection mechanisms such as TPE-optimized Self-Attention BiLSTM to achieve end-to-end resilience by combining proactive attack detection with adaptive scheduling.

Author Contributions: Conceptualization, F.K. and A.B.; methodology, F.K. and A.B.; software, F.K.; validation, F.K. and A.B.; formal analysis, F.K.; investigation, F.K.; resources, F.K.; data curation, F.K.; writing—original draft preparation, F.K.; writing—review and editing, F.K. and A.B.; visualization, F.K.; supervision, A.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zhang, A.-N.; Chu, S.-C.; Song, P.-C.; Wang, H.; Pan, J.-S. Task scheduling in cloud computing environment using advanced Phasmatodea population evolution algorithms. *Electronics* **2022**, *11*, 1451. [CrossRef]
2. Zhou, R.; Zeng, Y.; Jiao, L.; Zhong, Y.; Song, L. Online and Predictive Coordinated Cloud–Edge Scrubbing for DDoS Mitigation. *IEEE Trans. Mob. Comput.* **2024**, *23*, 9208–9223. [CrossRef]
3. Zhou, M.; Mu, X.; Liang, Y. SOE: A Multi-Objective Traffic Scheduling Engine for DDoS Mitigation with Isolation-Aware Optimization. *Mathematics* **2025**, *13*, 1853. [CrossRef]
4. Abdel-Basset, M.; Mohamed, R.; Abd Elkhalik, W.; Sharawi, M.; Sallam, K.M. Task Scheduling Approach in Cloud Computing Environment Using Hybrid Differential Evolution. *Mathematics* **2022**, *10*, 4049. [CrossRef]
5. Chandrashekar, C.; Krishnadoss, P.; Kedalu Poornachary, V.; Ananthakrishnan, B.; Rangasamy, K. HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Appl. Sci.* **2023**, *13*, 3433. [CrossRef]
6. Amer, D.A.; Attiya, G.; Ziedan, I. An efficient multi-objective scheduling algorithm based on spider monkey and ant colony optimization in cloud computing. *Clust. Comput.* **2024**, *27*, 1799–1819. [CrossRef]
7. Parthasaradi, V.; Karunamurthy, A.; Hussaian Basha, C.H.; Senthilkumar, S. Efficient task scheduling in cloud computing: A multiobjective strategy using horse herd–squirrel search algorithm. *Int. Trans. Electr. Energy Syst.* **2024**, *2024*, 1444493. [CrossRef]
8. Abraham, O.L.; Bin Ngadi, M.A.; Mohamad Sharif, J.B.; Mohd Sidik, M.K. Multi-objective optimization techniques in cloud task scheduling: A systematic literature review. *IEEE Access* **2025**, *13*, 54321–54345. [CrossRef]
9. Cui, M.; Wang, Y. An effective QoS-aware hybrid optimization approach for workflow scheduling in cloud computing. *Sensors* **2025**, *25*, 4705. [CrossRef]
10. Nandagopal, M.; Manavalan, T.; Praveen Kumar, K.; Manogaran, N.; Kesavan, D.; Al-Khasawneh, M.A. Enhancing energy efficiency in cloud computing through task scheduling with hybrid cuckoo search and transformer models. *Discov. Comput.* **2025**, *28*, 199. [CrossRef]
11. Khaleel, M.; Safran, M.; Alfarhood, S.; Gupta, D. Combinatorial metaheuristic methods to optimize the scheduling of scientific workflows in green DVFS-enabled edge–cloud computing. *Alex. Eng. J.* **2024**, *86*, 458–470. [CrossRef]
12. Long, G.; Wang, S.; Lv, C. QoS-aware resource management in cloud computing based on fuzzy meta-heuristic method. *Clust. Comput.* **2025**, *28*, 276. [CrossRef]
13. Dahan, F. An innovative approach for QoS-aware web service composition using whale optimization algorithm. *Sci. Rep.* **2024**, *14*, 22622. [CrossRef]
14. Mangalampalli, S.; Karri, G.R.; Kose, U. Multi-objective trust-aware task scheduling algorithm in cloud computing using whale optimization. *J. King Saud. Univ. Comput. Inf. Sci.* **2023**, *35*, 791–809. [CrossRef]
15. Naruei, I.; Keynia, F. Wild horse optimizer: A new meta-heuristic algorithm for solving engineering optimization problems. *Eng. Comput.* **2021**, *37*, 5095–5144. [CrossRef]
16. Saravanan, G.; Neelakandan, S.; Ezhumalai, P.; Maurya, S. Improved wild horse optimization with levy flight algorithm for effective task scheduling in cloud computing. *J. Cloud Comput.* **2023**, *12*, 24. [CrossRef]
17. Kaplan, F.; Babalık, A. Performance analysis of cloud computing task scheduling using metaheuristic algorithms in DDoS and normal environments. *Electronics* **2025**, *14*, 1988. [CrossRef]
18. Li, B.; Li, J.; Jia, M. ADFCNN-BiLSTM: A Deep Neural Network Based on Attention and Deformable Convolution for Network Intrusion Detection. *Sensors* **2025**, *25*, 1382. [CrossRef]
19. Chandrasiri, S.; Meedeniya, D. Energy-Efficient Dynamic Workflow Scheduling in Cloud Environments Using Deep Learning. *Sensors* **2025**, *25*, 1428. [CrossRef] [PubMed]
20. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
21. Peng, Z.; Pirozmand, P.; Motevalli, M.; Esmaeili, A. Genetic algorithm-based task scheduling in cloud computing using MapReduce framework. *Math. Probl. Eng.* **2022**, *2022*, 4290382. [CrossRef]
22. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks (ICNN), Perth, WA, Australia, 27 November–1 December 1995; Volume 1–6, pp. 1942–1948.
23. Lipsa, S.; Dash, R.K. SLA-based task scheduling in cloud computing using randomized PSO algorithm. In Proceedings of the Symposium on Computing and Intelligent Systems, New Delhi, India, 10 May 2024; pp. 206–217.
24. Karaboga, D. *An Idea Based on Bee Swarm for Numerical Optimization (Technical Report TR06)*; Department of Computer Engineering, Erciyes University: Kayseri, Turkey, 2005; pp. 1–10.

25. Li, J.; Han, Y. A hybrid multi-objective artificial bee colony algorithm for flexible task scheduling problems in cloud computing system. *Clust. Comput.* **2020**, *23*, 2483–2499. [CrossRef]
26. Askarzadeh, A. A novel metaheuristic method for solving constrained engineering optimization problems: Crow Search Algorithm. *Comput. Struct.* **2016**, *169*, 1–12. [CrossRef]
27. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl.-Based Syst.* **2016**, *96*, 120–133.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.