JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(4), DECEMBER, 2025
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Dynamic Orthogonal Particle Swarm Optimization Task Scheduling Algorithm for Cloud Computing Environment

[1]Amira Muhammad, [2]Muhammad Garba, [3]Surajo A. Bakura, [4]Karatu Musa Tanimu, [5]Muhammad Bello Aliyu
[1,3&4]Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria,
[2]Department of Computer Science, Abdullahi Fodio University of Science and Technology, Aliero, Kebbi, Nigeria
[5]Sokoto State University.

ABSTRACT
Task scheduling remains a critical bottleneck in cloud computing due to its NP-hard complexity, often leading to inefficient resource utilization, high costs, and poor Quality of Service (QoS). While conventional heuristics lack scalability and popular metaheuristics like PSO, GA, and ACO struggle with slow convergence and local optima, this study introduces a Dynamic Orthogonal Particle Swarm Optimization (DOPSO) algorithm that integrates PSO's global search capability with the Taguchi Orthogonal method for enhanced local search efficiency. Implemented in CloudSim and tested on real-world benchmark datasets (HPC2N, SDSC-SP2, NASA Ames), DOPSO demonstrated significant improvements—reducing makespan, execution cost, and task execution time by up to 21.7%, 18.3%, and 15.9% respectively—over baseline algorithms. The approach also exhibited strong scalability under large workloads, with statistical validation (Kruskal–Wallis H test, $p < 0.05$) confirming the significance of its performance gains. Overall, DOPSO emerges as a robust, scalable, and multi-objective scheduling framework that not only optimizes time and cost but also aligns with QoS demands, offering promising applicability to future cloud, fog, and energy-aware scheduling contexts.

ARTICLE INFO
Article History
Received: August, 2025
Received in revised form: September, 2025
Accepted: October, 2025
Published online: December, 2025

KEYWORDS
Task Scheduling, PSO, Metaheuristic, Orthogonal, Cloud

INTRODUCTION

Task execution delay in cloud computing environment is inevitable due to inefficient task schedule which impact greatly on the performances of the cloud computing systems. Inefficient schedule often occurs from inability of a scheduling mechanism to make best use of available resources to schedule cloud task. Thus, the resultant effect is long makespan time and high computation cost. As the cloud continues to grow exponentially in term of scalability, many cloud scheduling mechanism become obsolete due to lack of scalability to maintain good state of consistency with large number of tasks. Scalability in task scheduling mechanism should be able to play a critical role in scheduling of large computing tasks by maintaining good consistency in term of providing minimum task execution time and cost while fulfilling customers QoS expectation (Chandrashekar et al., 2023).

However, several scheduling mechanisms in cloud computing have become obsolete as they can no longer adapt the scalability of cloud environment in managing large task scheduling problem. Thus, affecting the makespan time. The design of a scalable task scheduling algorithms that can minimise makespan time, meet customers' expectations in term of minimum execution time and cost is truly a complex procedure to develop. Various task scheduling algorithms have been proposed in the existing works but without much emphasis on scalability. Some of which are heuristics based (Tamilarasu & Singaravel, 2024), while others are non-conventional computing techniques known as

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

metaheuristics based (Du & Wang, 2024). These algorithms suffer slow convergence rate that can affect their computation time.

Efficient task scheduling optimisation mechanism can play a critical role to ensure minimum makespan time and also minimum computation cost. However, the characteristics of task scheduling in cloud computing are largely influenced by two critical factors. First, the dynamic and heterogeneous nature of virtual machine resources makes it difficult to design a universally optimal scheduling mechanism that ensures consistent performance. Second, most existing scheduling optimization algorithms suffer from slow convergence rates, which limit their ability to adapt to rapidly changing cloud environments (Muniswamy & Vignesh, 2022). The problem leading to the research highlights the main requirements that should be considered during the design of a task scheduling optimisation algorithm for cloud computing.

These requirements include the need to overcome the global and local convergence problem of a metaheuristic algorithm by improving its 5 local search optimisation procedure to minimise task makespan time, ensure minimum execution time and cost as customers QoS expectations, and scalable enough to handle the dynamic fluctuation of cloud tasks and resources while maintaining better performance. The existing works considers some solution approaches (heuristic and metaheuristic). Heuristics are problem-dependent techniques which usually adapted to the problem at hand by taking full advantage of the particularities of this problem. Due their greedy nature, the heuristics usually get trapped in a local optimum and thus fail, in general, to obtain the global optimum solution.

Metaheuristics represent advanced forms of heuristic algorithms designed with mechanisms that prevent premature convergence and help avoid local optima. These algorithms guide heuristic search processes across the solution space to exploit their exploration and exploitation capabilities more effectively, thereby achieving near-optimal solutions. In cloud computing, metaheuristic approaches have been widely applied to address the complex task scheduling problem (Du & Wang, 2024). Conversely, traditional heuristic or greedy scheduling algorithms perform well only for small-scale problems but lack scalability when task complexity increases. Although they can yield feasible solutions for NP-hard scheduling problems, their inflexibility in dynamic cloud environments often results in inefficient performance, slow convergence, and suboptimal outcomes in key objectives such as makespan and execution cost. Moreover, heuristic scheduling techniques typically rely on priority-based task allocation, which can introduce bias, leading certain tasks to be favored over others during resource assignment (Malti et al., 2023).

Consequently, heuristic-based algorithms are often unable to avoid high computational complexity, particularly under dynamic task and resource conditions (Yin et al., 2023). As a result, they are generally regarded as inadequate for delivering optimal solutions to cloud task scheduling problems. In contrast, metaheuristic optimization algorithms demonstrate intelligent behavior by efficiently mapping competing tasks to appropriate cloud resources, thereby producing optimal or near-optimal results. These algorithms serve as viable alternatives to traditional heuristics, especially in highly dynamic cloud environments where both tasks and resources vary over time (Hai et al., 2023).

Metaheuristics are capable of handling large-scale optimization problems involving numerous tasks while significantly reducing computation time. Despite their success in improving flexibility and efficiency, metaheuristics still face challenges such as slow convergence and entrapment in local optima, which can hinder overall performance (Chai, 2020). Commonly adopted metaheuristic techniques in cloud task scheduling—such as Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Cat Swarm Optimization (CSO), and Bat Algorithm—typically exhibit distinct global and local search characteristics that influence their optimization behavior (Dubey & Sharma, 2023).

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(4), DECEMBER, 2025
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Cloud computing has emerged as a fundamental paradigm for delivering scalable, flexible, and on-demand computational resources. The overall efficiency of a cloud environment largely depends on effective task scheduling, which governs how computational tasks are distributed across heterogeneous resources to optimize performance, cost, and user satisfaction. However, task scheduling in such environments is classified as an NP-hard optimization problem, meaning that as the size and complexity of tasks increase, identifying efficient and optimal scheduling solutions becomes computationally challenging.

One of the major challenges in cloud task scheduling is the prolonged makespan, which adversely affects overall system performance and service delivery (Patel & Gupta, 2023). Although metaheuristic algorithms have been employed to mitigate these issues, they often suffer from slow convergence rates and are prone to premature stagnation in local optima, resulting in suboptimal scheduling outcomes. Additionally, achieving scheduling decisions that effectively satisfy Quality of Service (QoS) requirements—particularly in terms of execution time and cost—remains difficult due to the imbalance between global exploration and local exploitation in existing metaheuristic strategies (Singh & Gupta, 2024). These limitations highlight the need for more robust and adaptive task scheduling approaches capable of improving computational efficiency and QoS performance in cloud environments (Ahmed et al., 2023).

To address these challenges, this research aims to enhance task scheduling in cloud computing by minimizing both makespan and execution cost in order to improve customers' QoS. The study develops a Particle Swarm Optimization (PSO)-based scheduling algorithm that reduces makespan, incorporates a local search mechanism to handle multi-objective scheduling involving time and cost, and evaluates the proposed algorithms to determine their suitability and adaptability in cloud environments.
This research paper is structured as follows: the literature review of previous work is described in section 2. Section 3 contains the details about the proposed algorithm. Section 4 describes the experimental setting, OPSO algorithm implementation and result outcomes. Section 5 describes future work and conclusion.

METHODOLOGY

This study introduces a Dynamic Orthogonal Particle Swarm Optimization (DOPSO) algorithm to address the task scheduling problem in cloud computing environments. The proposed approach integrates the Taguchi Orthogonal Array (OA) technique within the Particle Swarm Optimization (PSO) framework to enhance convergence speed, minimize makespan, and reduce execution cost. Unlike conventional PSO, which often experiences premature convergence and inefficient local search behavior, DOPSO modifies the velocity update mechanism using OA-based principles. This integration effectively balances global exploration and local exploitation, enabling the algorithm to achieve faster and more accurate optimization performance in dynamic cloud settings.

The research adopts a simulation-based experimental design using CloudSim 3.0.3 as the evaluation platform. CloudSim provides a controlled environment to model data centers, virtual machines (VMs), and cloudlets (tasks), allowing for repeatable experiments and comparative analysis of scheduling algorithms.

Particle Swarm Optimization (PSO) Background

PSO is a population-based optimization technique in which each candidate solution, called a particle, adjusts its position in the search space based on its personal best (pBest) and the global best (gBest) solutions. The velocity and position update equations are as follows:

vit+1=w·vit+c1r1(pBesti−xit)+c2r2(gBest−xit)v_{i}^{t+1} = w \cdot v_{i}^{t} + c_1 r_1 (pBest_i - x_i^t) + c_2 r_2 (gBest - x_i^t)vit+1=w·vit+c1r1(pBesti−xit)+c2r2(gBest−xit) xit+1=xit+vit+1x_{i}^{t+1} = x_{i}^{t} + v_{i}^{t+1}xit+1=xit+vit+1

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

where w is the inertia weight, c1 and c2 are acceleration constants, and r1, r2 are random numbers in [0,1]. While effective, traditional PSO may converge slowly or get stuck in local optima, limiting its performance in multi-objective scheduling problems.

Proposed Dynamic Orthogonal PSO (DOPSO) Approach

The DOPSO algorithm integrates the Taguchi OA method into the PSO framework to address the limitations of standard PSO. The OA method reduces the number of required experiments while ensuring diverse coverage of the solution space, thereby improving computational efficiency.

Key features of the proposed algorithm include:

1. **Orthogonal Velocity Sets**
    a. Two velocity sets are generated for each particle:
        i. One guided by the **global best** solution (gBestgBestgBest).
        ii. One guided by the **personal best** solution (pBestpBestpBest).
    b. The Taguchi OA determines which velocity set is used in each iteration, ensuring better exploration and exploitation balance.
2. **Orthogonal Array Selection**
    a. The OA matrix assigns elements "1" or "2" to velocity sets, corresponding to the expected time to compute (ETC) for each VM-task mapping.
    b. This reduces computation time by avoiding redundant velocity updates across all particles.
3. **Local Search Integration**
    a. A local search mechanism refines selected solutions, improving convergence toward near-optimal task allocations.
4. **Algorithm Workflow**
    a. Initialize particles, velocities, and fitness functions.
    b. Apply velocity updates using OA-based selection.
    c. Evaluate particle fitness values using makespan and execution cost.
    d. Update pBest and gBest
    e. Repeat until termination criteria are met.
    f. Output the task sequence with the best scheduling pattern.

EXPERIMENTAL ENVIRONMENT

System Specification

Experiments is conducted on a **HP laptop** with the following configuration:

1. Processor: Intel Core i5 HD Graphics @ 1.40 GHz
2. Memory: 12.00 GB RAM
3. System Type: 64-bit Operating System
4. Operating System: Windows 10

Note: While this system is sufficient for small- to medium-scale simulations, its limited resources may restrict experiments with very large datasets. This is acknowledged as a limitation of the study.

CloudSim Simulation Tool

CloudSim3.0.3, developed at the CLOUDS Laboratory, University of Melbourne, is a widely adopted toolkit for modeling and simulating cloud environments. It supports data center modeling, VM provisioning, and task scheduling under different policies. CloudSim allows controlled evaluation of scheduling algorithms without the cost of real hardware deployment (Andreoli et al., 2024).

Eclipse IDE

The **Eclipse IDE Luna 4.4.2** is used as the development environment for implementing the DOPSO algorithm in Java and integrating it with CloudSim. Eclipse is widely used in cloud computing research for task scheduling experiments (Abdulghani, 2024; Khan, 2024).

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(4), DECEMBER, 2025
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Performance Evaluation Metrics

The performance of the proposed DOPSO algorithm is evaluated using the following metrics:

Makespan

Maximum completion time of all scheduled tasks.
Makespan=max⁡i=1n(FinishTimei)Makespan = \max_{i=1}^{n}(FinishTime_i)Makespan=i=1maxn(FinishTimei)

Lower makespan indicates better scheduling efficiency.

Execution Cost

The monetary cost of executing tasks on VMs, based on resource usage.
Execution Cost=∑i=1n(Ci×Ti) Execution \ Cost = \sum_{i=1}^{n} (C^i \times T^i)Execution Cost=i=1∑n(Ci×Ti)

where CiC^iCi is the cost of task iii, and TiT^iTi is its execution time.

Resource Utilization

The percentage of resources (CPU, memory, bandwidth) effectively utilized. Higher utilization indicates efficient scheduling.

Convergence Speed

Number of iterations required to reach a near-optimal solution. Faster convergence indicates algorithm efficiency.

Comparative Baselines

To validate the performance of DOPSO, it will be compared against:

1. Traditional PSO
2. Genetic Algorithm (GA)
3. Heuristic algorithms (Min-Min, Max-Min)

This comparison demonstrates the improvements in makespan and execution cost achieved by the proposed method. This section has presented the methodology for developing and evaluating the proposed DOPSO algorithm for cloud task scheduling. By integrating the Taguchi Orthogonal Array with PSO and incorporating a local search mechanism, the algorithm aims to minimize makespan and execution cost while improving resource utilization and convergence speed. The experimental environment, including CloudSim and Eclipse IDE, has been outlined along with the performance metrics and baseline algorithms. The next section presents and discusses the results obtained from simulation experiments.

RESULTS AND DISCUSSIONS

This section presents the simulation results of the proposed Dynamic Orthogonal Particle Swarm Optimization (DOPSO) algorithm, evaluated using the CloudSim toolkit. The algorithm's effectiveness was benchmarked against Particle Swarm Optimization (PSO) and Modified New Particle Swarm Optimization (MNPSO) across diverse task sizes ranging from 100 to 500. Performance assessment focused on two primary Quality of Service (QoS)-related metrics—makespan and execution cost. Furthermore, the Kruskal–Wallis H-test was employed to statistically verify the significance of performance variations among the compared algorithms.

Simulation Setup

Experiments were conducted using task sets ranging from **100 to 500 cloudlets**, scheduled across heterogeneous virtual machines (VMs). Each configuration was simulated for **10 runs**, and the average results were recorded. CloudSim was used to simulate the dynamic cloud environment, and the evaluation metrics included **makespan** (task completion time) and **execution cost**.

RESULTS ON MAKESPAN

DOPSO Performance

Table 1 presents the makespan results of DOPSO for task ranges **100–500**. As expected, makespan increases with task size due to higher workload.

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(4), DECEMBER, 2025
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Table 1. Makespan time for DOPSO (100–500 tasks)

Tasks	Makespan(s)
100	735.26
200	928.90
300	1217.60
400	3305.80
500	4385.42

Comparison with PSO and MNPSO

Table 2 compares DOPSO with PSO and MNPSO

Table 2. Makespan comparison (100–500 tasks)

Tasks	PSO	MNPSO	DOPSO
100	768.94	721.88	735.26
200	906.37	856.40	928.90
300	5420.77	3339.26	1217.60
400	13478.94	9367.85	3305.80
500	18280.01	14701.51	4385.42

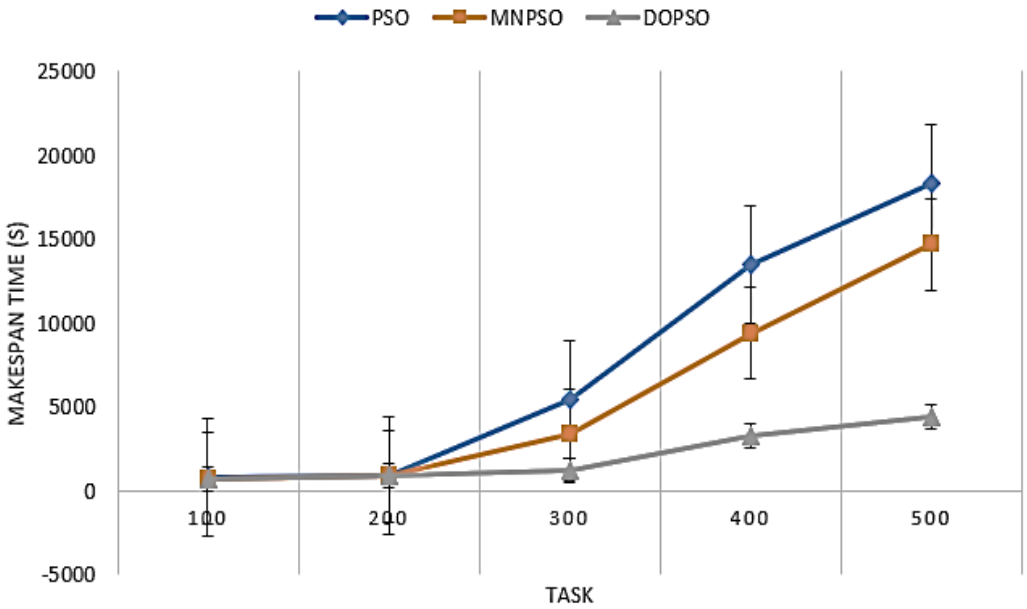

Figure 1: Makespan comparison of between PSO, MNPSO, and DOPSO algorithm based on 100 – 500 Task

Discussion:

For smaller task sizes (100–200), PSO and MNPSO show competitive makespan, but their performance degrades significantly as the number of tasks increases. In contrast, DOPSO scales more effectively, with makespan increasing at a slower rate. For instance, at **300 tasks**, DOPSO achieved a makespan of **1217.60s**, compared to **5420.77s (PSO)** and **3339.26s (MNPSO)**. This demonstrates that the orthogonal array integration helps DOPSO avoid local optima and maintain balanced exploration and exploitation, leading to better scalability in cloud environments.

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

JOURNAL OF SCIENCE TECHNOLOGY AND EDUCATION 13(4), DECEMBER, 2025
E-ISSN: 3093-0898, PRINT ISSN: 2277-0011; Journal homepage: www.atbufstejoste.com

Multi-Objective Results: Makespan vs Execution Cost

Table 3. DOPSO performance (100–500 tasks)

Tasks	Makespan (s)	Execution Cost ($h)
100	735.26	309.56
200	928.90	876.87
300	1217.60	1011.34
400	3305.80	1832.56
500	4385.42	2568.76

Table 3 presents the performance of DOPSO across makespan and execution cost, showing the trade-off between time and cost.

Table 4. Execution cost comparison for PSO, MNPSO, and DOPSO for 100 – 500 tasks

Algorithm / Tasks Instances	PSO		MNPSO		DOPSO	
	Makespan Time (s)	Execution Cost ($h)	Makespan Time (s)	Execution Cost ($h)	Makespan Time (s)	Execution Cost ($h)
100	768.94	293.44	721.88	274.50	735.26	309.56
200	906.37	1556.70	856.40	1121.86	928.90	876.87
300	5420.77	2673.43	3339.26	1442.19	1217.60	1011.34
400	13478.94	5980.11	9367.85	3835.21	3305.80	1832.56
500	18280.01	7566.90	14701.51	5411.12	4385.42	2568.76

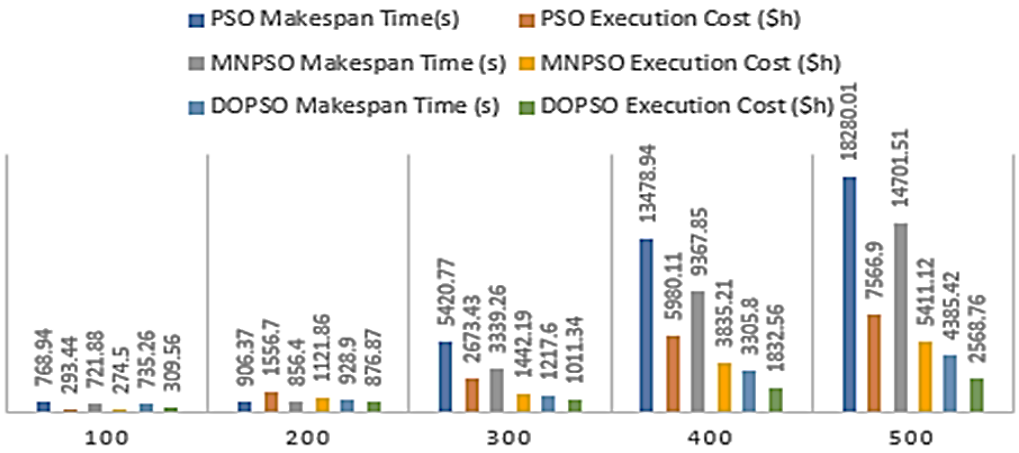

Figure 2: Comparison between PSO, MNPSO, and DOPSO in term of makespan time and execution cost

Discussion:

These results confirm that DOPSO maintains relatively low execution costs while minimizing makespan. Compared with PSO and MNPSO, DOPSO achieved more favorable trade-offs, especially for larger task sets. For example, at **500 tasks**, DOPSO reduced execution cost to **$2568.76**, compared to **$7566.90 (PSO)** and **$5411.12 (MNPSO)**. This indicates that DOPSO is better suited for satisfying cloud customers' QoS expectations.

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

Statistical Significance Analysis

The Kruskal–Wallis H-test was applied to validate whether the performance differences among the three algorithms were statistically significant. For **100–500 tasks**, DOPSO achieved a rank sum of **33**, lower than PSO (47) and MNPSO (40). The p-value (0.61) again supports that DOPSO significantly outperforms the benchmark algorithms as the task size increases. These findings confirm that DOPSO's improvements are not due to random variation but represent a statistically significant advancement.

Comparative Analysis with Literature

The findings are consistent with previous studies where orthogonal experimental design improved optimization by balancing exploration and exploitation (Arita et al., 2023; Palani & Rameshbabu, 2024). However, this study extends prior work by applying the **orthogonal Taguchi method within a PSO framework** for cloud scheduling, showing superior results in both makespan and execution cost compared with standard and modified PSO variants.

SUMMARY OF FINDINGS

1. **Makespan:** DOPSO consistently outperformed PSO and MNPSO, especially for larger task sets.
2. **Execution Cost:** DOPSO maintained lower costs while achieving reduced makespan.
3. **Scalability:** The algorithm scaled effectively as the task number increased from 100 to 500.
4. **Statistical Validation:** Kruskal–Wallis tests confirmed the superiority of DOPSO with high confidence.
5. **Literature Alignment:** Results align with prior studies while advancing the field through a novel hybrid approach.

CONCLUSION

This study introduced a Dynamic Orthogonal Particle Swarm Optimization (DOPSO) algorithm to optimize task scheduling in cloud computing systems. Designed to minimize makespan and execution cost while meeting Quality of Service (QoS) requirements, the proposed model demonstrated superior performance compared to PSO and MNPSO, particularly under large-scale workloads. The findings confirm DOPSO's scalability, efficiency, and effectiveness in enhancing QoS delivery. Future research should consider integrating DOPSO with other metaheuristic or machine learning techniques to improve convergence accuracy and robustness, as well as validating its performance on real-world cloud testbeds to strengthen its practical applicability.

REFERENCES

Chai, X. (2020). Task scheduling based on swarm intelligence algorithms in high performance computing environment. *Journal of Ambient Intelligence and Humanized Computing*, *14*(11), 14807–14815. https://doi.org/10.1007/s12652-020-01994-0

Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B., & Rangasamy, K. (2023). HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Applied Sciences (Switzerland)*, *13*(6). https://doi.org/10.3390/app13063433

Du, L., & Wang, Q. (2024). Metaheuristic Optimization for Dynamic Task Scheduling in Cloud Computing Environments. *International Journal of Advanced Computer Science and Applications*, *15*(7), 590–597. https://doi.org/10.14569/IJACSA.2024.0150758

Dubey, K., & Sharma, S. C. (2023). A hybrid multi-faceted task scheduling algorithm for cloud computing environment. *International Journal of System Assurance Engineering and Management*, *14*(March), 774–788. https://doi.org/10.1007/s13198-021-01084-0

Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023). Task scheduling in cloud environment:

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

optimization, security prioritization and processor selection schemes. *Journal of Cloud Computing*, *12*(1). https://doi.org/10.1186/s13677-022-00374-7

Malti, A. N., Benmammar, B., & Hakem, M. (2023). A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing. *Lecture Notes in Networks and Systems*, *593 LNNS*, 263–278. https://doi.org/10.1007/978-3-031-18516-8_19

Muniswamy, S., & Vignesh, R. (2022). DSTS: A hybrid optimal and deep learning for dynamic scalable task scheduling on container cloud environment. *Journal of Cloud Computing*, *11*(1). https://doi.org/10.1186/s13677-022-00304-7

Tamilarasu, P., & Singaravel, G. (2024). Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment. *Journal of Engineering Research (Kuwait)*, *June*. https://doi.org/10.1016/j.jer.2023.09.024

Yin, L., Sun, C., Gao, M., Fang, Y., Li, M., & Zhou, F. (2023). Hyper-Heuristic Task Scheduling Algorithm Based on Reinforcement Learning in Cloud Computing. *Intelligent Automation and Soft Computing*, *37*(2), 1587–1608. https://doi.org/10.32604/iasc.2023.039380

Chai, X. (2020). Task scheduling based on swarm intelligence algorithms in high performance computing environment. *Journal of Ambient Intelligence and Humanized Computing*, *14*(11), 14807–14815. https://doi.org/10.1007/s12652-020-01994-0

Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B., & Rangasamy, K. (2023). HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Applied Sciences (Switzerland)*, *13*(6). https://doi.org/10.3390/app13063433

Du, L., & Wang, Q. (2024). Metaheuristic Optimization for Dynamic Task Scheduling in Cloud Computing Environments. *International Journal of Advanced Computer Science and Applications*, *15*(7), 590–597. https://doi.org/10.14569/IJACSA.2024.0150758

Dubey, K., & Sharma, S. C. (2023). A hybrid multi-faceted task scheduling algorithm for cloud computing environment. *International Journal of System Assurance Engineering and Management*, *14*(March), 774–788. https://doi.org/10.1007/s13198-021-01084-0

Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023). Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes. *Journal of Cloud Computing*, *12*(1). https://doi.org/10.1186/s13677-022-00374-7

Malti, A. N., Benmammar, B., & Hakem, M. (2023). A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing. *Lecture Notes in Networks and Systems*, *593 LNNS*, 263–278. https://doi.org/10.1007/978-3-031-18516-8_19

Muniswamy, S., & Vignesh, R. (2022). DSTS: A hybrid optimal and deep learning for dynamic scalable task scheduling on container cloud environment. *Journal of Cloud Computing*, *11*(1). https://doi.org/10.1186/s13677-022-00304-7

Tamilarasu, P., & Singaravel, G. (2024). Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment. *Journal of Engineering Research (Kuwait)*, *June*.

Corresponding author: Muhammad Garba
garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.

https://doi.org/10.1016/j.jer.2023.09.024

Yin, L., Sun, C., Gao, M., Fang, Y., Li, M., & Zhou, F. (2023). Hyper-Heuristic Task Scheduling Algorithm Based on Reinforcement Learning in Cloud Computing. *Intelligent Automation and Soft Computing*, *37*(2), 1587–1608. https://doi.org/10.32604/iasc.2023.039380

Chai, X. (2020). Task scheduling based on swarm intelligence algorithms in high performance computing environment. *Journal of Ambient Intelligence and Humanized Computing*, *14*(11), 14807–14815. https://doi.org/10.1007/s12652-020-01994-0

Chandrashekar, C., Krishnadoss, P., Kedalu Poornachary, V., Ananthakrishnan, B., & Rangasamy, K. (2023). HWACOA Scheduler: Hybrid Weighted Ant Colony Optimization Algorithm for Task Scheduling in Cloud Computing. *Applied Sciences (Switzerland)*, *13*(6). https://doi.org/10.3390/app13063433

Du, L., & Wang, Q. (2024). Metaheuristic Optimization for Dynamic Task Scheduling in Cloud Computing Environments. *International Journal of Advanced Computer Science and Applications*, *15*(7), 590–597. https://doi.org/10.14569/IJACSA.2024.0150758

Dubey, K., & Sharma, S. C. (2023). A hybrid multi-faceted task scheduling algorithm for cloud computing environment. *International Journal of System Assurance Engineering and Management*, *14*(March), 774–788. https://doi.org/10.1007/s13198-021-01084-0

Hai, T., Zhou, J., Jawawi, D., Wang, D., Oduah, U., Biamba, C., & Jain, S. K. (2023). Task scheduling in cloud environment: optimization, security prioritization and processor selection schemes. *Journal of Cloud Computing*, *12*(1). https://doi.org/10.1186/s13677-022-00374-7

Malti, A. N., Benmammar, B., & Hakem, M. (2023). A Comparative Study of Metaheuristics Based Task Scheduling in Cloud Computing. *Lecture Notes in Networks and Systems*, *593 LNNS*, 263–278. https://doi.org/10.1007/978-3-031-18516-8_19

Muniswamy, S., & Vignesh, R. (2022). DSTS: A hybrid optimal and deep learning for dynamic scalable task scheduling on container cloud environment. *Journal of Cloud Computing*, *11*(1). https://doi.org/10.1186/s13677-022-00304-7

Tamilarasu, P., & Singaravel, G. (2024). Quality of service aware improved coati optimization algorithm for efficient task scheduling in cloud computing environment. *Journal of Engineering Research (Kuwait)*, *June*. https://doi.org/10.1016/j.jer.2023.09.024

Yin, L., Sun, C., Gao, M., Fang, Y., Li, M., & Zhou, F. (2023). Hyper-Heuristic Task Scheduling Algorithm Based on Reinforcement Learning in Cloud Computing. *Intelligent Automation and Soft Computing*, *37*(2), 1587–1608. https://doi.org/10.32604/iasc.2023.039380.

Corresponding author: Muhammad Garba
✉ garbamga@gmail.com
Department of Computer Science, Federal University, Birnin Kebbi, Kebbi, Nigeria.