Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

Fault-Tolerant Task Scheduling in Multicore Systems Using Hybrid Metaheuristic Algorithms

[1]Adu, Folashade Christiana, [2]Igiri C.G, [3]Ogbolotuo Imumesen Solomon, [4]Ezekiel Cookey

[1,2,3,4]Rivers State University, Nkpolu, Oroworukwu, Rivers State, Nigeria
E-mail: [1]xtiana1202@gmail.com, [2]igiri.Chima@ust.edu.ng, [3]imumesensolomon@gmail.com, [4]ezekiel.cookey@ust.edu.ng

Abstract: Ensuring reliable and efficient task scheduling remains a critical challenge in multicore computing environments, particularly when system faults can significantly affect performance and interfere with execution. This paper presents a hybrid optimization strategy that combines Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) techniques to improve task allocation under fault-prone conditions. The proposed model considers task dependencies during scheduling and dynamically distributes workloads across available processing cores to achieve balanced utilization while maintaining reliability.To evaluate its effectiveness, the hybrid GA–PSO method was tested against standalone GA and PSO approaches. The experimental findings indicate that the combined strategy achieves shorter execution times, better scalability as workload increases, and a noticeable reduction in task failure rates. These results suggest that integrating evolutionary and swarm-based optimization mechanisms can provide a practical and robust solution for improving both performance and fault tolerance for modern multicore systems.

Keywords: Fault-tolerant, Scheduling; Multicore Computing; Hybrid Optimization; Genetic Algorithm; Particle Swarm Optimization; Task Allocation; Scalability; System Reliability.

I. Introduction

Recent developments in multicore processor technology have greatly improved the computational capacity of modern computing systems. By integrating multiple processing units within a single architecture, multicore systems enable parallel task execution and higher throughput. However, the same characteristics that improve performance also introduce new reliability challenges. Hardware ageing, transient faults, thermal stress, and unpredictable runtime behavior may disrupt task execution and negatively affect system performance. As a result, reliability has become a critical design consideration alongside performance optimization.

Task scheduling in multicore environments focuses on assigning interdependent tasks to available processor cores while optimizing metrics such as execution time and processor utilization. When reliability considerations are included, the scheduling problem becomes a multi-objective optimization task that simultaneously addresses performance efficiency, system availability, and fault tolerance. Due to the combinatorial complexity of mapping tasks to processors, the problem is generally classified as NP-hard.

Consequently, exhaustive search techniques are impractical for large systems, and heuristic or metaheuristic optimization strategies are typically used to produce near-optimal solutions within reasonable computational time.

In distributed computing environments, fault-tolerant scheduling mechanisms play an important role in minimizing the impact of processor failures and improving the dependability of the overall system (Khan *et al.*, 2023). At the same time, metaheuristic algorithms such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) have become widely adopted for solving complex scheduling problems because they can efficiently explore large solution spaces and adapt to dynamic conditions (Zhang & Qi, 2021).

This study proposes a hybrid scheduling approach that integrates GA and PSO techniques to improve scheduling reliability and execution efficiency in multicore systems. The proposed framework combines the global search ability of GA with the fast convergence behavior of PSO. In addition, selective task replication is incorporated to enhance fault tolerance under unreliable runtime conditions. The major contribution of this work is a scalable hybrid scheduling strategy that simultaneously improves execution performance and system reliability while controlling computational overhead.

Typical objective functions considered in fault-tolerant multicore scheduling include:

1. Minimizing overall execution time (makespan)

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

2. Reducing energy consumption
3. Maximizing system reliability
4. Minimizing redundancy and replication overhead

Main contributions of this study include:

1. Development of a hybrid GA–PSO scheduling framework for multicore systems.
2. Integration of selective task replication for improved fault tolerance.
3. Evaluation of the hybrid algorithm against standalone GA and PSO approaches.
4. Demonstration of improved scheduling efficiency and reliability.

II. Related Work

Scheduling in distributed and parallel computing systems has been widely studied using various optimization approaches. Evolutionary algorithms and swarm intelligence methods are among the most popular techniques used to solve complex scheduling problems.

Genetic Algorithms are well known for their ability to perform global searches across large solution spaces. However, they may require many iterations before reaching high-quality solutions, especially when scheduling problems involve complex dependency constraints. On the other hand, Particle Swarm Optimization often converges more quickly but may become trapped in local optima when the search space is highly complex (Zhang & Qi, 2021).

To address these limitations, several researchers have explored hybrid optimization approaches that combine the strengths of different algorithms. Studies indicate that hybrid metaheuristic techniques often outperform single optimization strategies in reliability-aware scheduling environments (Khan *et al.*, 2023). For example, hybrid frameworks integrating evolutionary operations with swarm intelligence have demonstrated improved convergence behavior in heterogeneous and distributed computing systems (Li *et al.*, 2021).

In addition, reliability-aware scheduling models have shown that incorporating failure probability into scheduling decisions can significantly reduce execution disruptions and improve system dependability (Xie *et al.*, 2019). Recent research also highlights the effectiveness of hybrid optimization techniques that combine Particle Swarm Optimization with other evolutionary methods in improving convergence speed and scheduling performance (Prasad *et al.*, 2025).

Although many scheduling techniques have been proposed, several studies focus mainly on optimizing execution time while treating reliability mechanisms separately. A unified scheduling framework that simultaneously improves execution efficiency and system reliability remains an important research challenge. This work addresses this gap by proposing a hybrid GA–PSO scheduling strategy that integrates dependency-aware scheduling with selective replication.

III. System Architecture

The proposed scheduling framework is organized as a layered system that processes incoming tasks, performs optimization, and allocates tasks to available processor cores. The architecture integrates evolutionary computation and swarm intelligence techniques to improve the efficiency of scheduling decisions. Hybrid optimization frameworks with similar designs have been applied successfully in distributed computing systems (Li *et al.*, 2021).

The main components of the architecture include:

1. Task Input Layer (Application Tasks)

- Receives computational tasks submitted by applications.
- Tasks are modeled using a Directed Acyclic Graph (DAG) to represent task dependencies.

2. Pre-processing Module (Task Analysis)

- Extracts important task attributes such as execution time, deadline, and priority level.
- Identifies dependency relationships among tasks in the DAG.

3. Genetic Algorithm Exploration Engine

- Generates an initial population of candidate scheduling solutions.
- Applies crossover and mutation operations to explore different task-to-core allocations.

4. PSO Optimization Engine

- Further refines promising solutions obtained from the GA phase.
- Updates particle positions and velocities to guide the search toward optimal scheduling configurations.

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

5. Fault Tolerance Module

- Implements selective replication for tasks identified as critical.
- Helps reduce the probability of execution failure.

6. Multicore Execution Layer

- Deploys the final optimized schedule across the available processor cores.
- Monitoring mechanisms track task completion time and detect execution failures.

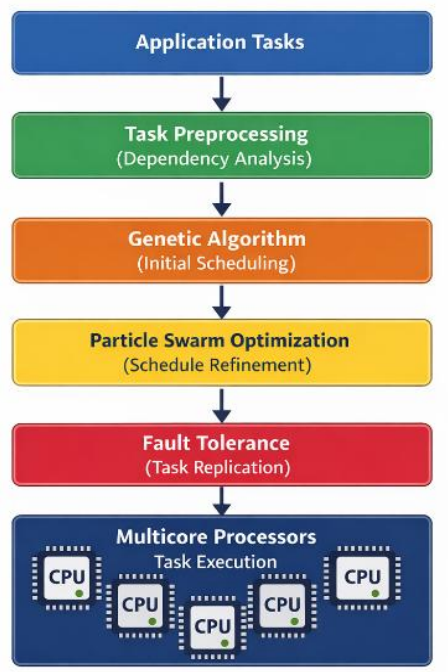

Figure 1: Architecture of the proposed Hybrid GA-PSO fault-tolerant scheduling framework. Source: Adapted from Li et al. (2022) and Zhang & Qi (2021)

The system begins with application tasks represented as a directed acyclic graph (DAG). The preprocessing stage analyzes task dependencies. The Genetic Algorithm generates candidate schedules using crossover and mutation operations. These solutions are refined using Particle Swarm Optimization. A fault-tolerance mechanism replicates critical tasks before execution on multicore processors.

IV. Proposed Hybrid GA–PSO Scheduling Approach

4.1 Problem model and notation

Let $T = \{t_1, t_2, \dots, t_n\}$ denote the set of tasks organised as a directed acyclic graph (DAG), where an edge $t_i \rightarrow t_j$ indicates that t_j cannot start until t_i completes. Let $M = \{M_1, M_2, \dots, M_m\}$ be the set of processing cores (or machines). A schedule is an assignment

$$\phi: T \rightarrow M,$$

subject to precedence constraints and resource capacity constraints. Each machine M_k is characterised by parameters such as processing speed s_k, failure rate λ_k, and availability A_k (homogeneous or heterogeneous models are both supported).

Each task t_i is described by: execution time ET_i, deadline D_i, reliability requirement R_i, and optional priority P_i.

4.2 Two-stage hybrid search

The scheduler operates in two stages:

Stage 1 (GA exploration): GA generates diverse candidate schedules using encoding of task-to-core mappings, crossover, and mutation. Diversity mitigates early convergence and broadens coverage of the feasible region under DAG constraints.

Stage 2 (PSO refinement): The best GA candidates seed PSO. Particle updates refine schedules using velocity-driven search to improve convergence speed and solution quality.

Hybrid algorithms improve on

1. Exploration refers to searching for new areas of the solution space to discover potentially better solutions e.g PSO
2. Exploitation is refining and improving the best solutions found so far e.g., GA.
3. Local refinement (e.g., Simulated Annealing)

Hybrid algorithms also optimize the composite objective function under constraints:

1. Task precedence constraints
2. Core availability
3. Deadline constraints
4. Fault-recovery constraints

4.3 Fault tolerance via selective replication

Fault tolerance is incorporated by replicating selected critical tasks, guided by reliability requirements R_i and observed failure characteristics of machines. Replication is applied selectively to limit overhead while reducing the probability of

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

task failure, particularly for tasks on critical paths.

4.4 Fitness Function

4.4.1 Objectives

The scheduler seeks to minimise makespan and reduce the failed task rate while maintaining feasibility with respect to precedence constraints. A weighted fitness function is adopted to enable GA and PSO to maximise a scalar quality score.

4.4.2 Normalisation and weighted fitness

Since execution time and failure rate are minimisation objectives, each is normalised to a maximisation scale using min–max inversion:

$$\text{Norm}(x) = \frac{x_{\text{worst}} - x}{x_{\text{worst}} - x_{\text{best}}}.$$

Using experimental bounds:

- $x_{\text{worst}}^{time} = 120\ s, x_{\text{best}}^{time} = 92\ s$
- $x_{\text{worst}}^{fail} = 8.5\%, x_{\text{best}}^{fail} = 2.9\%$

The weighted fitness is defined as:

$$F = 0.6 \cdot \text{Norm(Time)} + 0.4 \cdot \text{Norm(Failure)},$$

where execution time receives higher weight due to its direct influence on throughput.

Normalised results (from your measurements) are summarised below:

Algorithm	Normalised Time	Normalised Failure	Fitness F
GA	0.00	0.00	0.00
PSO	0.36	0.41	≈ 0.38
Hybrid GA–PSO	1.00	1.00	1.00

These values indicate that the hybrid GA–PSO provides the best balance between performance and fault tolerance under the evaluated settings.

V. Experimental Evaluation

5.1 Simulation Environment

The proposed scheduling algorithm was evaluated using simulation tools commonly employed in distributed computing research (Buyya *et al.*, 2002). Two simulation environments were utilized:

- CloudSim, which supports modeling of cloud infrastructures, virtual machines, and scheduling policies.
- SimGrid, a toolkit designed for simulating large-scale distributed computing environments and evaluating scheduling algorithms.

The experimental setup included 100 tasks, while the number of processor cores ranged from4 to 16. The performance of GA, PSO, and the hybrid GA-PSO algorithm was compared using metrics such as execution time and task failure rate.

Simulation platforms allow researchers to evaluate scheduling algorithms in a controlled environment without requiring expensive physical infrastructures (Casanova *et al.*, 2008).

5.2 Experimental setup

Simulation-based experiments were conducted to compare GA, PSO, and hybrid GA–PSO for dependency-constrained scheduling under fault-prone conditions. Evaluation metrics include average execution time, failed-task percentage, and scalability as the number of cores increases, consistent with established multicore scheduling evaluation practice (Zhang, *et al* 2021).

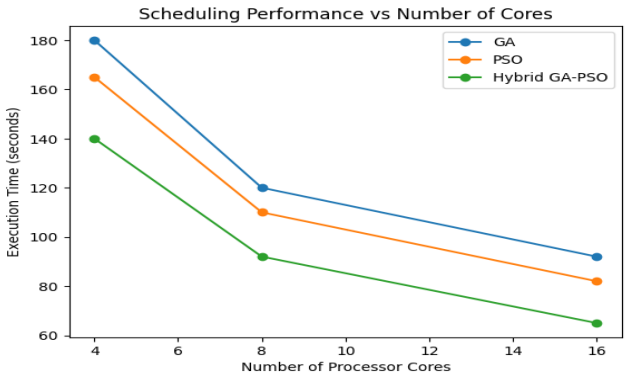

Figure 2: Execution time comparison of GA, PSO, and Hybrid GA-PSO algorithms as the number of processor cores increases. Source: Adapted from Jacob & Pradeep (2021)

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

The results show that execution time decreases as the number of processor cores increases. However, the hybrid GA-PSO algorithm consistently achieves lower execution time compared to standalone GA and PSO approaches, demonstrating improved scalability and efficiency.

VI. Results and Discussion

6.1 Execution Time Performance

Experimental results indicate that execution time decreases as the number of available processor cores increases. The hybrid GA–PSO approach consistently achieves lower execution times compared with standalone GA and PSO methods.

6.2 Reliability Evaluation

The hybrid GA–PSO achieves the lowest makespan and the smallest failure rate, indicating improved throughput and higher scheduling reliability.

Table 1: Performance and reliability comparison

Algorithm	Average Execution Time (s)	Failed Tasks (%)	Reliability Level
GA	120	8.5	Medium
PSO	110	6.2	Medium-High
Hybrid GA–PSO	92	2.9	High

The table shows when scheduling 100 tasks, GA fails 8-9 tasks on average, PSO fails about 6, and hybrid GA-PSO fails only 3.

6.3 Scalability Analysis

Performance was evaluated using different processor counts.

Table 2: Scalability experiment

Number of Cores	GA (s)	PSO (s)	Hybrid GA–PSO (s)
4	180	165	140
8	120	110	92
16	92	82	65

The hybrid algorithm demonstrates the strongest performance improvement as system resources increase. As shown in the table, when using 8 processor cores, GA takes 120 seconds to complete all tasks. PSO takes 110 seconds, but the hybrid GA-PSO only takes 92 seconds.

The experimental results demonstrate that hybrid metaheuristic techniques generally outperform single optimization algorithms in scheduling problems due to their enhanced exploration and convergence capabilities (Jacob & Pradeep, 2021). The proposed algorithm improves scheduling efficiency by combining the exploration strength of GA with the rapid convergence of PSO.

Through this integration, the scheduler is able to avoid local optima and generate more reliable task-to-core allocation strategies. Previous comparative studies also indicate that population-based optimization algorithms often achieve better performance than traditional scheduling techniques when dealing with complex workflow scheduling problems (Subramoney & Nyirenda, 2021). The key findings of this work are:

1. The hybrid scheduling approach significantly reduces overall execution time compared with standalone GA and PSO methods.
2. Selective replication improves fault tolerance by reducing the number of failed tasks and increasing system reliability.
3. The hybrid algorithm maintains stable performance as the number of processor cores increases.

6.4 Implementation and Applications

The proposed scheduling framework can be applied in several computing environments, including:

- Cloud computing platforms
- Large-scale data centers
- High-performance computing clusters
- Embedded systems requiring reliable execution

6.5 Advantages

1. Improved fault tolerance through selective replication.
2. Reduced execution time relative to standalone GA and PSO.
3. Strong scalability with increasing core count.
4. Reduced risk of premature stagnation via hybrid exploration–exploitation.
5. Practical and easy-to-implement framework

Journal of Artificial Intelligence and Emerging Technologies
Volume 3, Issue 5, pp 17-22, May-2026
ISSN (online): 3049-1967
DOI: https://doi.org/10.47001/JAIET/2026.305002
Available Online at https://www.jaiet.com

6.6 Limitations

1. Higher computational overhead because both GA and PSO algorithms are combined.
2. Performance depends on appropriate parameter tuning, such as population size and learning coefficients.
3. Task replication increases the use of system resources.
4. Experimental validation is limited to simulation environments; performance in real systems may vary.

VII. Conclusion

Reliable task scheduling is essential for maintaining performance in multicore computing systems that may experience hardware faults or runtime disruptions. This study introduced a hybrid GA–PSO scheduling framework that integrates evolutionary and swarm-based optimization techniques. The approach combines global search capability with rapid convergence and incorporates selective task replication to improve reliability.

Simulation results demonstrate that the hybrid approach achieves reduced execution time, lower task failure rates, and improved scalability compared with standalone GA and PSO algorithms. These findings suggest that hybrid metaheuristic techniques represent a promising solution for reliability-aware scheduling in modern parallel computing systems.

Future work will focus on evaluating the framework in real distributed computing environments and exploring energy-aware scheduling strategies.

REFERENCES

[1] Buyya, R., & Murshed, M. (2002). GridSim: A toolkit for modeling and simulation of distributed resource management and scheduling.

[2] Casanova, H., Legrand, A., & Quinson, M. (2008). SimGrid: A generic framework for large-scale distributed experiments.

[3] Jacob, I., & Pradeep, S. (2021). Multi-objective task scheduling using hybrid particle swarm optimization in cloud computing.

[4] Khan, S., Liu, P., & Abbas, H. (2023). Fault-tolerant scheduling in multicore systems: A survey. *ACM Computing Surveys*, 55(2).

[5] Kumar, A., & Sharma, P. (2019). Cost-efficient scheduling in cloud environments using hybrid meta heuristic algorithms.

[6] Li, Y., Chen, Z., & Sun, F.(2022).Hybrid GA-PSO algorithms for task scheduling. *Future Generation Computer Systems*, 125, 456–468.

[7] Prasad, R., Roy, A., & Kumari, S. (2025). Hybrid PSO-GWO approach for efficient cloud task scheduling.

[8] Subramoney, D., & Nyirenda, C. (2020). Comparative evaluation of population-based optimization algorithms for workflow scheduling.

[9] Xie, L., Yang, Y., & Li, J. (2019).Genetic algorithm-based task scheduling for multicore processors. *Journal of Systems Architecture*, 96, 10–21.

[10] Zhang, H., & Qi, X. (2021). Particle swarm optimization for multicore task scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 32(4), 876–887.

Citation of this Article:

Adu, Folashade Christiana, Igiri C.G, Ogbolotuo Imumesen Solomon, & Ezekiel Cookey. (2026). Fault-Tolerant Task Scheduling in Multicore Systems Using Hybrid Metaheuristic Algorithms. *Journal of Artificial Intelligence and Emerging Technologies (JAIET).* 3(5), 17-22. Article DOI: https://doi.org/10.47001/JAIET/2026.305002

***** End of the Article *****