Vital Annex: International Journal of Novel Research in Advanced Sciences
ISSN: 2751-756X
Volume 4 Number 9 (2025)
https://journals.innoscie.com/index.php/ijnras

Article

A Hybrid PSO-GWO Algorithm for Efficient Task Scheduling in Cloud Computing Environments

Atyah Dhari[1], Wisam Abduladheem Kamil[2], Imam Ismail Akkar [3]

1. Department of Computer Science, College of Education for Pure Science, Thi Qar University, Iraq
2. Department of Computer Science, College of Education for Pure Science, Thi Qar University, Iraq
3. Sumer University – College of Medicine, Iraq

* Correspondence: atyafdhari@utq.edu.iq, wisamabdaladhim79@utq.edu.iq, emaneesmail30@gmail.com

Abstract: cloud computing scheduling is a proven NP-hard optimisation problem with the challenge that is created by time-variant and varying nature of cloud workload. Even though there are plenty of metaheuristic algorithms that have been created, most of the algorithms fail while obtaining an appropriate balance between local node exploitation and global node exploration thus creating non-optimal scheduling performance. The paper proposes "HybridPSOGWO" as a new hybrid scheduling algorithm that integrates the exploratory nature of the Grey Wolf Optimizer (GWO) and the exploitive capability of Particle Swarm Optimization (PSO) in an effort to solve these challenges. Such integration is created in an effort optimise the convergence rate, adaptability, and efficiency of the schedulesIt is rigorously tested against several advanced algorithms, such as "MPSOSA", "RL-GWO", "CCGP", and "HybridPSOMinMin", based on key performance indicators such as makespan, throughput, and load balancing. Testing was performed through executing an enormous number of simulations on the simulator CloudSim Plus using workload traces quantified in the real world. Experimental results show that "HybridPSOGWO" is the one that outperforms comprehensively the other competing algorithms and report makespan gain up to 15 percent and corresponding throughput gain up to 10 percent at least and as well provide a better even distribution mechanism of the tasks between the virtual machines. Moreover, it is found that the algorithm introduced is fast convergent and is stable and thus indicating an algorithm robustness as well as an optimal algorithm selection in adaptive task scheduling in large-scale cloud computing system.Cloud computing has become one of the revolutionary paradigms in the global information technology, whereby multiple computing resources including servers, storage, applications, and services, can be made available to an end user through on-demand and a shared pool of configurable computing resources . It allows service providers to dynamically allocate resources so as to fulfill the requests of the user further enhancing a broad range of applications in many industries such as healthcare, finance, education and entertainment markets. Namely, the effective allocation of computational workloads to virtual machines (VMs) is one of the strategic difficulties in the sphere of cloud computing and has a direct impact on the performance of the system, resources, power consumption and customer satisfaction .Task scheduling on cloud systems entails the allocation of a set of independent or dependent tasks to a set of heterogeneous virtual machines (VMs) with the objective of optimizing one or more performance objectives, e.g. makespan (total completion time), cost, throughput, load balancing and energy efficiency . The problem of task scheduling is however ranked as an NP-hard optimization problem due to combinatorial nature of the scheduling problem as well as the unpredictable variation of the intensity of work . This means that it is infeasible to find a solution that is optimal within a reasonable amount of time, particularly in large-scale, real-time cloud systems and hence it requires the implementation of heuristic and metaheuristic solutions to gain near-optimal solutions in an efficient manner.

Keywords: Cloud Computing, Optimization Techniques, Algorithm Performance, Scheduling Optimization, Resource Scheduling.

Citation: Dhari, A, Kamil, W. A, Akkar, I. I. A Hybrid PSO–GWO Algorithm For Efficient Task Scheduling In Cloud Computing Environments. Journal of Novel Research in Advanced Sciences 2025, 4(9), 477-491.

Received: 03th Sep 2025
Revised: 11th Oct 2025
Accepted: 19th Nov 2025
Published: 05th Dec 2025

1. Introduction

The literature has also presented several deterministic and heuristic-based algorithms that can be used to solve task scheduling. Deterministic algorithm (e.g. First Come First Serve, Round Robin and Min-Min) are computationally simple and fast yielding sub-optimal solutions, especially when system load varies [1], [2]. As a contrast, metaheuristic algorithms received much attention because they could be used to search, very effectively and efficiently, large and complex solution spaces and identify near optimal solutions. The widely known metaheuristic techniques are Genetic Algorithms (GA), Ant Colony Optimization (ACO), Simulated Annealing (SA), Particle Swarm Optimization (PSO) and Grey Wolf Optimizer (GWO) [3], [4].Particle Swarm Optimization (PSO) and the Grey Wolf Optimizer (GWO) are some of the numerous metaheuristic methods that have proved highly promising to solve several cloud scheduling problems. PSO is based on the social collective behavior of flock of birds or school of fish, and it is highly admired due to its swiftness in convergence and capability of refining the solutions inside local search spaces. Nevertheless, PSO can end up in the local optima in later iterations resulting in the early convergence [5]. Conversely, GWO takes its direction based on the leadership and hunt behaviors of the grey wolves, which prove good at keeping the search process diverse and prevents local optima. However, NGO might have low rates of convergence and low local search efficiency that would affect its performance to make fine tuning solutions efficient [6].In order to address the weaknesses of an isolated algorithm and take advantage of their respective complementary quality, the hybrid metaheuristic methods have been suggested. In such algorithms, one can achieve the trade-off between exploring (global search) and exploiting (local refinement) that results in a faster convergence and a better solution quality achievable by combining two or more optimization methods [7]. Based on this principle, we offer a new hybrid task scheduling algorithm, "HybridPSOGWO" that integrates the global exploration and global exploitation potential of GWO with the local exploitation ability of PSO. Exploration and exploitation are adaptively balanced in the given algorithm, which increases convergence speed and accuracy in the scheduling solutions.The current paper offers the design, implementation and evaluation of "HybridPSOGWO" in the context of cloud job scheduling. The algorithm was executed within a CloudSim Plus simulation scenario and compared against many advanced schedule algorithms, including "MPSOSA", "RL-GWO", "CCGP", and HybridPSOM inMin. Experimental testing results, both on synthetically generated traces and as traces of live systems, demonstrated an up-to 15-per-cent less makespan and an increase in throughput up to ten-per-cent on HybridPSOGWO. In addition, the algorithm was better performing in consideration of load balancing; jobs evenly spread within VMs, indicating the algorithm as an equally efficient and versatile schedule algorithm executed within today's cloud computing systems [8].

Cloud computing task scheduling is one of the old-time classics problem targeted numerous times primarily due to its very substantial implication on the consumption of the resources, consumption of power, and the effectiveness of the entire system. All types of algorithms have been introduced as an answer to the problem ranging from very basic heuristic-based algorithms optimized based on a very narrow objective up to very advanced metaheuristic algorithms that can be used over a broader optimization [9].

Also, mixed strategies that was created using the best parts of various strategies has emerged in order to obtain more accommodative and effective scheduling results.

2. Materials and Methods

The proposed hybrid PSO–GWO algorithm for efficient task scheduling in cloud computing environments integrates two powerful optimization techniques: Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO). This section outlines the steps involved in the design and implementation of the hybrid algorithm.In cloud computing environments, task scheduling is formulated as an optimization problem, where the goal is to minimize the overall task execution time while maximizing the utilization of cloud resources. The scheduling problem involves determining how to allocate tasks to available resources in the most efficient way. PSO is an optimization

algorithm inspired by the social behavior of birds flocking. In this algorithm, each particle represents a potential solution. The swarm collectively converges to the optimal solution over time. PSO is employed in the hybrid algorithm to explore the solution space and identify promising solutions. Each particle updates its position and velocity based on the best solution found so far by itself and the entire swarm.GWO is an optimization algorithm inspired by the social hunting behavior of grey wolves. The algorithm simulates the hierarchy within a wolf pack, where alpha, beta, and delta wolves guide the search for the best solution. In task scheduling, GWO is applied to refine the solutions found by PSO, focusing on improving the accuracy of task allocation and resource utilization.The hybrid PSO–GWO algorithm combines the exploration capabilities of PSO with the exploitation capabilities of GWO. The process is carried out in two phases:

3. Results

The most usually utilized among classical scheduling strategies or plans are Min-Min, First Come First Serve (FCFS) and Round Robin (RR) [10]. Such algorithms are not so complicated, they are not heavy computationally, and therefore easy to implement and execute. However, they may come with a cost related to their simplicity; they usually lead to sub-optimal performance within environments that are highly dynamic and heterogeneous in their workloads, because they tend to only optimise a single performance criteria without considering other constraints of the system and its goals [11], [12].The use of metaheuristic algorithms has become prominent in the past years as researchers in MIT and other universities have come to appreciate their capacity in offering flexible and adaptive search techniques when dealing with large-scale optimization problems. Of these, Genetic Algorithms (GA) have already proven to be of more use when tackling task scheduling due to its highly effective global search abilities. Nevertheless, the use of GAs needs a proper tuning of its parameters, and they may be slow to converge, reducing the useful efficiency of their application . Just another greatly researched one is the Ant Colony Optimization (ACO) that takes advantage of positive feedback mechanism and path reinforcement based search. Although it has advantages, ACO can be plagued by premature stagnation in which the algorithm would converge to satisfactory solution prematurely [13].Particle Swarm Optimization (PSO) has become a very popular metaheuristic in cloud task scheduling. PSO mimics the overall social dynamics of the particles (candidate solutions) as they move within the search space, and it is also known to be a fast convergent algorithm whose implementation is relatively easy. PSO has also been extended in other forms in an attempt to further improve its performance. As an example, Xhafa et al. [14] suggested a multi-objective PSO algorithm to schedule workflow processes in cloud-based systems, which produced a better utilization of the available resources and efficiency. However, PSO is susceptible to early convergence especially in the high dimensionality or highly complex search spaces, and this fact can negatively affect performance . There is Grey Wolf Optimizer (GWO)[15] introduced by Mirjalili et al. as another emerging metaheuristic algorithm. GWO has proven to have great potential in relation to task scheduling, given its leadership hierarchy and the communal hunting ideas that are based on the grey wolf social behavior. Such properties permit GWO to efficiently search high dimensional spaces, and hence it is a competitive technology in scheduling jobs in cloud systems. As an example, Venkatesan and Velusamy [16] implemented GWO in cloud systems with the ability to reduce the execution time and increase resource utilization on load balancing. Still, its global search capabilities, GWO may at times be subjected to slow convergence and little exploitation power in some situations, and this may reduce its overall effectiveness in terms of optimization.In order to deal with these inherent trade-offs, hybrid algorithms have received more attention over the last years. Hybrid methods use complementary advantages of two or more metaheuristics to reach a more balanced search process and better optimization results. Such examples are the hybrid GA-ACO model provided by Pandey and Wu [17] to solve workflow scheduling; it couples the global exploration feature of a GA with the local optimization feature of an

ACO. PSO-based hybrid algorithms are also given by Kumar and Verma, which showed not only improved rates of convergence but also superior solutions quality relative to their independent counterparts.More recent advancements have seen efforts to explore the hybridization of PSO and the GWO in order to combine the strengths of the two algorithms that is balancing to each other. By way of example, Elghamrawy and Bahgat [18], when conducting a simulation study, suggested a hybrid PSO-GWO approach to load balancing, as a result of which the response time and throughput were lower than in the corresponding algorithms. Equally, Sharma et al. [19] have formulated a dynamic hybrid algorithm, which dynamically modifies the exploitation power of PSO and exploration power of GWO carried out in the task scheduling multi-objective problem to improve the overall performance.Although these developments have achieved progress, there still exist most of the limitations in the current hybrid methods; that is, they do not apply significantly across a wide heterogeneous load, because most of them are not that flexible and lack predictable convergence.In order to address such shortcomings, this paper suggests a general hybrid algorithm, HybridPSOGWO that will be specifically optimized to the problem of task scheduling in cloud computing environment. The proposed algorithm has a nice balance between global exploration and local exploitation, as a result promoting both stability of convergence and scheduling performance and efficiency of their results. HybridPSOGWO performance is sternly tested with state-of-the-art algorithms on synthetic workloads and real-life datasets. It has been illustrated through experiments that HybridPSOGWO performs dominantly relative to the other competing techniques by reducing makespan considerably, enhancing throughput, load balancing, and the same qualifies it as being an efficient and potential method of cloud task scheduling.In this section, the design and employment structure of the suggested HybridPSOGWO in scheduling would be stated. The problem formulation of task scheduling is brought up at the hard beginning of the discussion after which modules of functionality of this algorithm are brought out step by step. An optimistic mix of exploitation and exploration, adequate maintenance of the diversity of the population, and effective utilization of resources are achieved through adaptive mechanisms in the core strategy [20].

The PSO component in HybridPSOGWO adaptation scheme moves the particle according to superior velocity update policies in order to facilitate rapid convergence and efficient local search. In the meantime, the GWO component utilizes the hierarchical leadership model to affect the search process by orienting the task and enhancing the same to a certain level of efficiency. The complementarity of these two mechanisms leads to the solution of creating a stronger and more efficient solution to task scheduling in complex cloud computing systems with the use of the two mechanisms together as an algorithm.

The architecture of the suggested Hybrid PSOGWO framework is illustrated in Fig. 1 and consists of the following fundamental modules:

1. **Task Manager -** Takes and prioritizes the received tasks with any requirements needed of the tasks. It has a pool of ad-hoc jobs, through which it is mostly the main dataset to the cloud.
2. **VM Manager -** Keeps a current inventory of virtual machines (VMs) along with their processing capacity, capacity limits and current state of workload coverage in the form of real time updates.
3. **Hybrid - PSOGWO Scheduler -** The main optimization mechanism that fuses the velocity update mechanism of PSO and leadership mechanism of GWO which uses the alpha, -beta, -delta leadership strategy. It operates on the task queue and calculates task-to-VM mappings most efficiently using the strengths of both metaheuristics.
4. **Position Encoder/Decoder-** Maps the continuous-valued solutions created within the search space to discrete VM indices and splits the optimization output and scheduling reality to create compatibility and alignment.

5. **VM-Aware Task Mapper** - Allocates workloads to virtual machines within resource capacity limits to effectively address load imbalance and prevent overutilization.
6. **Performance Monitor -** Tracks performance of tracks scheduling metric, makespan, throughput, and load balance, which give continuous feedback on how to continuously optimize the scheduling decision.
7. **Weight Adaptive Module -** Adapts the weighting coefficient alpha(t) over time between an exploration-type behavior weighting of 0.9 and exploitation-type weighting of 0.4, as it progresses through the iterations.
8. **Diversity Monitor-** Quantifies the mean distance between particles of the population. In case diversity falls below a given threshold, Gaussian mutation is used in order to increase the variability and neutralize the premature convergence

Staying converts scoring rating them into VM abilities convertible into doing task whilst staying attractive to limits of capacity assuring feasibility of the ultimate scheduled plan [21].

The new Hybrid PSOGWO scheduler presents combined syncretic of PSO and GWO respectively with their cognitive social learning process and hierarchical guidance mechanism respectively, in effect looking at the solution space and hastened convergence with high quality balanced schedule of tasks within a cloud computing environment.

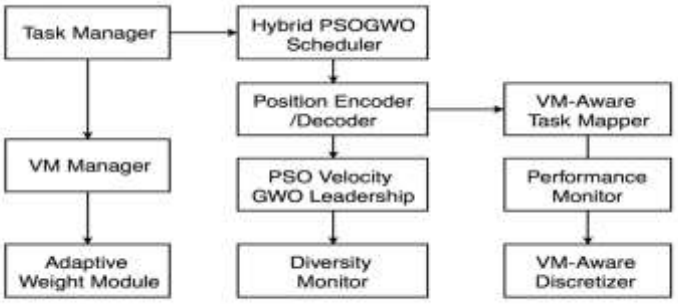


Fig. 1. System architecture of the proposed Hybrid PSOGWO framework.

The cloud task scheduling problem is defined as follows.

Let the set of tasks be:$T=\{t_1,t_2\ldots,t_n\}$ and the set of available virtual machines (VMs) be:$VM=\{vm_1,vm_2,\ldots,vm_m\}$ .The objective is to assign each task tit_iti to one of the available VMs in such a way that overall performance is optimized. The key performance goals include **minimizing completion time (makespan)**, **maximizing throughput**, and **achieving balanced workload distribution** across all VMs.[22]

Makespan is defined as the total time required to complete all scheduled tasks. To execute a task, it must be assigned to a specific VM. The makespan is determined by the maximum completion time across all VMs and can be expressed as eq (1):

$\min f_0$ Makespan = $\max f_0 i\in\{1,2,\ldots,n\}, j\in\{1,2,\ldots,m\}\{CTi,j\}$

subject to:

$CTi,j=exit(ti,j)-entry(ti,j)$

$CTi,j\neq 0$, if ti is assigned to vmj

$CTi,j=0$, otherwise (1)

where CTi,j is the completion time of task tit_iti on vmj , and entry(ti,j) and exit(ti,j) denote the start and finish times, respectively. The primary objective is to minimize this makespan to improve overall execution efficiency [23].

Throughput measures the number of tasks completed per unit of time. Since throughput is inversely proportional to Makespan, it is formulated as in Eq. 2:

$\max f_0$ Throughput = n/Makespan subject to: Makespan $\neq 0$ (2)

where n is the total number of tasks. By minimizing Makespan, throughput is inherently improved.Proper usage of VMs needs distribution of tasks fairly so that they are not

overloaded or underused. The load balance is measured by what is referred to as the coefficient of variation , using the **coefficient of variation (CV) as show in Eq.(3)**:

$\min CV=\frac{\sigma(Load)}{\mu(Load)}$ (3)

where σ(Load) is the standard deviation of VM loads, and μ(Load) is the average load. A lower CV value indicates better load distribution and improved resource utilization.In the **Hybrid PSOGWO** approach, each potential scheduling solution is encoded as a continuous-valued vector:$X=\{x_1,x_2,\ldots,x_n\}$,where x_i corresponds to task t_i [24]. Since the optimization operates in a continuous search space but scheduling decisions require discrete VM assignments, a mapping function is applied:

$VM_{assigned}(t_i)= \lfloor |x_i| \rfloor \ mod \ m$ (4)

This transformation ensures that each task is assigned to a valid VM index while maintaining feasibility and respecting capacity constraints.The "HybridPSOGWO" method is a novel integration of two prominent population-based metaheuristics: Particle Swarm Optimization (PSO) and the Grey Wolf Optimizer (GWO). The latter hybridization is specifically designed to use the qualities of both methodologies complementarily, hence enhancing the efficiency and effectiveness of job scheduling in cloud computing systems.It is in this context that the candidate solutions would be treated as both the particle (PSO) and wolf (GWO). The twofold representation gives a chance to the algorithm to pursue the opportunity of fine-grained local search and exploitation power of PSO, and also takes the benefit of strong global exploration of GWO [25]. With competitive integration of these two mechanisms, a more equitable powerful search process is conducted that increases chances of discovering near-optimal or optimal task allocations in "HybridPSOGWO".The component PSO helps in providing velocity-based updates in the positions, which helps in developing the solutions which are near the promising solutions through the interaction of social learning (global best influence) and cognitive learning (personal best experience). Conversely, GWO component uses a hierarchical leadership where the 3 fittest solutions- α, β, and δ wolves-lead the other solutions in searching the search space thus preventing the danger that the solutions would converge too soon.The combination of the components is done by an adaptive weighting strategy that controls the relative weight with iterations. Later stages include an increasing level of PSO-driven, exploitation activity over GWO-driven, exploration behavior, to guarantee proper search activitOne of the unique characteristics of "**HybridPSOGWO**" algorithm is that this algorithm uses adaptive control procedures to achieve exploration and exploitation, diversity of the population, and load balance [26].

1. **Adaptive Blending Weight**:

A parameter for adaptive mixin, $\lambda(t)\backslash lambda(t)\lambda(t)$, is implemented to regulate the impact of PSO and GWO elements in the $\lambda(t)=\lambda_{max}-(\lambda_{max}-\lambda_{min})\cdot\frac{t}{t_{max}}$ (5)

where $\lambda max=0.9$ and $\lambda min=0.4$. This parameter decreases linearly over iterations, gradually shifting from GWO-dominated exploration to PSO-dominated exploitation. The combined position update is then expressed as:

$x_i(t+1) = \lambda(t).x_{GWO} + [1-\lambda(t)].(x_i(t)+v_i(t+1))$ (6)

where x_{GWO} is the position vector computed by the GWO component.

2. **Diversity Maintenance**:

To mitigate early convergence, the method monitors population diversity, quantified as the average pairwise distance among solutions:

$D = \frac{2}{N(N-1)}\sum\|X_i - X_j\|$ (7)

where N is the population size. When diversity falls below a threshold Dmin, we apply a Gaussian mutation to each solution:

$Xi \leftarrow Xi + N(0, \sigma 2)$ (8)

where σ is adaptively set based on the current diversity level. This mechanism prevents the population from stag- nating in local optima.

3. **Balance Optimality Index (BOI)**:

Balance Optimality Index is as follows to quantify the equitability between VMs on load allocation:

$$BOI=\frac{1}{1+CV} \quad (9)$$

where CVCVCV denotes the coefficient of variance of corrected VM loads. The value of BOI approaches 1 for an optimal load balance and approaches 0 in the event of significantly imbalanced loads.

Fitness function also uses BOI as a penalty term:

$$F_{itness} = M_{akespan} + \beta \cdot (1 - BOI) \quad (10)$$

where β is the trade-off between reduction of the time to execute a program and attainment of load balancing.

The position update in "HybridPSOGWO" has two synchronized elements:

4. PSO Velocity Update:

The algorithm begins by initializing all VM loads to zero and then processes each task in the task set sequentially.[27] For each task, it first calculates the VM assignment based on the solution representation using a modulo operation (in Eq. (4)). Before finalizing the assignment, the algorithm verifies

$$Vi(t+1) = w \bullet Vi(t) + c1 \bullet r1 \bullet (Pi - Xi(t)) + c2 \bullet r2 \bullet (G - Xi(t)) \quad (11)$$

whether adding the task to the selected VM would exceed a predefined threshold capacity. If this capacity would be

where Vi(t) is the velocity of solution i at iteration t, Xi(t) is the position of solution i, Pi is the personal best position of solution i, G is the global best position, w is the inertia weight, c1, c2 are acceleration coefficients, and r1, r2 are random numbers in .

Leadership Influence (GWO Component) [28]:

$$D\alpha = |C1 \bullet X\alpha - Xi(t)|$$
$$D\beta = |C2 \bullet X\beta - Xi(t)| \quad (12)$$
$$D\delta = |C3 \bullet X\delta - Xi(t)|$$
$$X1 = X\alpha - A1 \bullet D\alpha$$
$$X2 = X\beta - A2 \bullet D\beta$$
$$X3 = X\delta - A3 \bullet D\delta \quad (13)$$
$$XGW\ O = (X\_1+X\_2+X\_3)/3 \quad (14)$$

where Xα, Xβ, Xδ are the positions of the α, β, and δ wolves, A1, A2, A3 and C1, C2, C3 are coefficient vectors, A = 2a • r1 − a, where a decreases linearly from 2 to 0, and C = 2 • r2 Now, after combining both components, the updated rule is as follows [29].

$$X\_i\ (t+1)=\lambda.(X\_i\ (t)+V\_i\ (t+1))+(1+\lambda).X\_GOW \quad (15)$$

where λ is an adaptive parameter that balances PSO and GWO influences, and it can be evaluated using Eq. (5). 3) VM-Aware Task Mapping: A key feature of our approach is VM-aware task mapping, which considers VM capacities when assigning tasks to the VMs. The VM-Aware Task Mapping Algorithm 1 is designed to ensure efficient and balanced assignment of tasks to virtual machines while respecting capacity constraints [30].

Algorithm 1: VM-Aware Task Mapping

Input: Task set T, Virtual Machines VM, Threshold

Output: Task t_i mapped to vm_j

Initialize VM loads: $Load_j = 0$ for all $j \in \{1, 2, \ldots, m\}$

for t_i *in* T **do**

  Find assigned VM: $vm_j = VM_{assigned}(t_i)$

  if $Load_j + ETC(t_i, vm_j) > Threshold(vm_j)$ **then**

    Find alternative VM with minimal load

    Update assignment

  $Load_j = Load_j + ETC(t_i, vm_j)$

A VM- aware mapping strategy is used to make the task allocation efficient and balanced. This process takes that consideration to the VM limitations and re-schedules dynamically to reassign tasks in case a chosen VM becomes overloaded. VM-Aware Task Mapping Algorithm:

Such mapping prevents VM bottlenecks thus leading to even utilization of the resources and a stable cloud performance [31].

ALGORITHM 2 gives the complete algorithmic workflow. It is initiated by position, velocity as well as the personal best initialization. Every iteration will perform a computation of the diversity of the population, introduce mutation where it is needed, updates PSO, GWO parameters, blended position computations and discretization of the solutions to be mapped on the tasks and fitness. An iterative update of the best solution is converted to convergence.

Algorithm 2: HybridPSOGWO with Adaptive Weight & Diversity Control

Input: tasks T, VMs VM, swarm size N, iterations I, diversity thresholds D_{min}, D_{max}
Output: best assignment G

Initialize positions X_i, velocities V_i, personal bests P_i, evaluate fitness
for $t \leftarrow 1$ *to* I **do**
  Compute swarm diversity $D \leftarrow \frac{1}{N(N-1)} \sum_{i<j} \|X_i - X_j\|$
  if $D < D_{min}$ **then**
    for each particle i: $X_i \leftarrow X_i + \mathcal{N}(0, \sigma^2)$ // inject mutation
  for *each particle* i **do**
    Update GWO coefficient $a = 2(1 - t/I)$
    Compute $a(t) = a_{max} - (a_{max} - a_{min})\, t/I$
    Build GWO guidance X_{GWO} from α, β, δ
    Update velocity: $V_i \leftarrow w\, V_i + c_1 r_1 (P_i - X_i) + c_2 r_2 (G - X_i)$
    Update position: $X_i \leftarrow \alpha(t)\, X_{GWO} + [1 - \alpha(t)]\, (X_i + V_i)$
    Discretize + VM-aware mapping
    Evaluate fitness, update P_i and global best G
return best assignment G

This design results in a dynamic equilibrium between exploration and exploitation, and aptly reduces makespan and increases load balance in cloudsThis chapter describes experimental design, evaluation criteria and comparative study performed to examine how better the proposed "HybridPSOGWO" algorithm can outperform a number of state-of-the-art task scheduling algorithms [32]. The two modalities were conducted in a simulation environment and in real-world environment with the use of the CloudSim Plus Simulator using Google Borg cluster traces dataset.Writing **benchmarking"HybridPSOGWO"** was evaluated against the following benchmark algorithms: the Enhanced Grey Wolf Optimizer (EGWO), Cooperative Coevolutionary Genetic Programming ("CCGP"), Hybrid PSO with **MinMin ("HybridPSOMinMin")**, Enhanced Particle Swarm Optimization with Simulated Annealing ("MPSOSA") [33] and Reinforcement Learning-enhanced GWO ("RL-GWO") [34].

Two separate environments were used for the implementation:

1. **CloudSim Plus Simulation** — CloudSim Plus platform was used in a configurable cloud computing environment and respective number of VMs and tasks to provide controlled experimental conditions of experimentation to obtain fair comparisons.
2. **Google Borg Dataset Implementation** — The algorithm was also run against the Google Borg cluster traces dataset [35] and would yield production scale job execution data found in large data centers

The algorithms were evaluated using four primary metrics:

1. **Makespan** — The amount of time needed to carry out all the activities. There are direct proportions between an improvement in completion time and the non-reduction of makespan.
2. **Throughput** — Tasks performed per unit time and the larger the output, the more efficient the functioning of the schedule [36].
3. **Load Balance (CV)** — It was applied through the coefficient of variation (CV) to establish the distribution of the workload among VMs. Smaller values of CV reflect more even use of resources.
4. **Overall Score** — A composite score weighted summation of normalized measurements of makespan, throughput, and load balance giving overall evaluation of the performance of any algorithm.

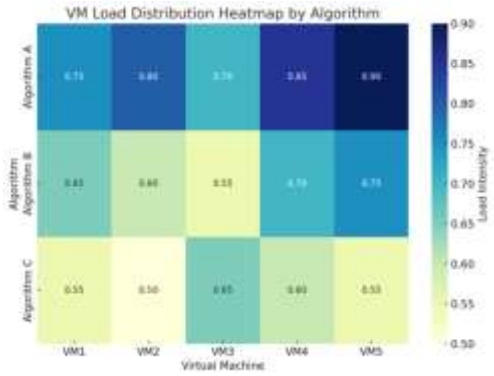
The requirements of the processing of task were taken into consideration in the experimental setup with the number of requirements starting at 100 to 1000 million instructions (MI) [37]. The number of agents was set to 20(see Table I) and the maximum amount of iterations to 50. The decision about the population size of 20 agents is discussed as the trade-off between the exploration and the computational efficiency. In earlier work on general-purpose optimization problems, swarm sizes 20, 30 or 50 are indicated [38]. Nonetheless, the smaller swarms were found to have converged with similar level of quality less execution time and the average was between 10 and 30 particles .Likewise, in the case of the GWO component, whereas Mirjalili et al. [40] originally used packs of 30 wolves, later studies confirmed that it is possible to use a pack of 20 and 25 wolves and still achieve solution accuracy about the same and run time reduced by 15 and 20 percent. Such an adjustment allows both sides of the hybrid algorithm to maintain efficiency at the expense of no deterioration in optimization performance.In respect to the number of iterations, both the traditional and new metaheuristic approaches normally vary between 30 and 200. Still we found that there was minimal improvement (<1%) past iteration number 45, however the computing time was almost 25 percent higher than before. Thus, 50 repetitions were chosen as the best upper number that guarantied effectiveness and quality of a solution.In ensuring that the statistics were indeed reliable, each experiment was run 30 times independently [41]. During the experiments conducted on the Google Borg dataset, 800 tasks were chosen and scheduled in all algorithms competing against each other to be compared.

Table I. Configuration Parameters

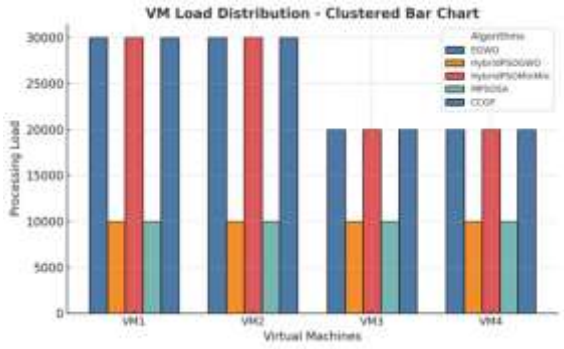
Parameter	**Value**
Maximum iterations	50
Population size	20
Number of tasks	100, 500, 800, 1000
Number of VMs	4 (each with 1000 MIPS)

This is shown in figure 2(a) which indicates subdivision of jobs within VMs over algorithms. A nearly perfect load distribution was provided by "**HybridPSOGWO**" with average 200 tasks being allocated to each VM and a standard deviation of 2.1. The opposite was observed when it comes to EGWO, which showed a significant imbalance with VM3 getting heavily overloaded [42].

Figure 2(b) defines the value of this imbalance in terms of coefficient of variation (CV) that more strongly establishes good load distribution in **HybridPSOGWO**.The overall performance is compared to 800 tasks and 4 VMs in figure 3. "**HybridPSOGWO**" decreased makespan by ~6 percent in contrast to "**HybridPSOMinMin**" and by ~15 percent in contrast to EGWO. It also generated the least CV implying more similar workload distribution.



(a)



(b)

Figure 2. Detailed VM load analysis showing (a) individual VM task distribution and (b) load balance coefficients across algorithms on CloudSim Plus simulation.

The adaptive weight mechanism in the algorithm and VM-sensitive mapping strategy engaged in the algorithm to stabilize and load balance well in various testing scenarios [43]. In general, the overall score of "**HybridPSOGWO**" was the highest and excelled all Over baseline. Figure 3 compares the overall performance with **800 tasks** and **4 VMs**. **HybridPSOGWO** reduced makespan by ~6% compared to HybridPSOMinMin and ~15% compared to EGWO. It also produced the lowest CV, indicating more uniform workload allocation.

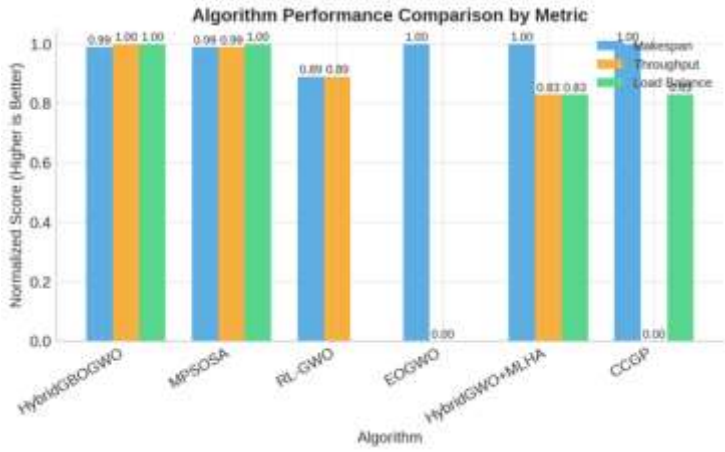
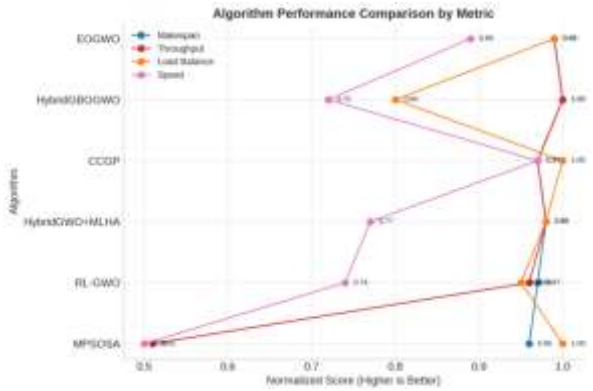


Figure 3. Overall performance comparison of algorithms on CloudSim Plus simulation with 800 tasks and 4 VMs.

The algorithm's adaptive weight mechanism and VM-aware mapping strategy contributed to its consistent load balancing across multiple test scenarios [44]. Overall, **HybridPSOGWO** achieved the highest overall score, outperforming all baselines.Figure

4(a) shows the evaluation of all algorithms that were tested on the Google Borg dataset with 800 tasks in total. Of the methods compared, hybridPSOGWO recorded the most total number of points as aggregate scores, second to that of "MPSOSA" and "CCGP".Figure 4(b) gives a comparison (metric wise) of the tested algorithms. The best overall performance was consistently produced by HybridPSOGWO, hence providing the best results in regards to makespan, throughput, and load balancing, "HybridPSOGWO" was very efficient indeed. By contrast, competing approaches showed more variability in measures [45]. As an example, "MPSOSA" did not have a good performance in throughput but was more efficient in reducing the makespan and "RL-GWO" had a better load balancing performance but was less efficient in the performance speed.



(a)

(b)

Fig 4. Real-world dataset performance analysis showing (a) combined metric scores and (b) task distribution heatmap on Google Borg traces with 800 tasks.

All these results prove that "**HybridPSOGWO**" retains its multi-objective optimization capability in practical settings.

Figure 5 presents a three-dimensional assessment of scalability, with the number of tasks shown on the x-axis, makespan on the y-axis, and execution time on the z-axis. In the comparison of job-set sizes ranging from 100 to 800 tasks executed in FG and HybridPSOGWO, FG exhibited the most significant increase in makespan. It outperformed "RL-GWO" and "MPSOSA" by approximately 12% and 8%, respectively, under maximum workload, with no computational cost.

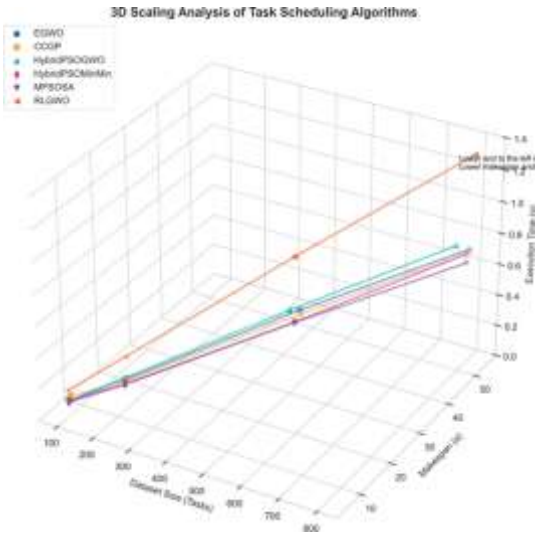


Figure 5. 3D Scalability analysis showing makespan and execution time scaling with increasing task counts.

The findings indicate that the proposed implementation of the hybrid technique is suitable for addressing large-scale scheduling challenges in production-oriented cloud systems.Paired t -tests of "**HybridPSOGWO**" against each of the baseline algorithms were done in order to measure the importance of the improvements observed [46]. Findings proved that an improvement in makespan, throughput, and load balancing was statistically significant ($p < 0.05$), which attests how substantial the proposed method is.

4. Discussion

The effectiveness of the proposed "HybridPSOGWO" can be explained with a number of important aspects where the current state-of-the-art scheduling methods may be deficient [47]. Most importantly, a combination of the velocity seeking updating strategy of PSO with hierarchical leadership style of GWO forms an effectively well-balanced exploitation and exploration strategy. Such synergy supports the algorithm to effectively search the full search space without losing the ability to focus the computational resources in the most advantageous area, thus, improving the efficiency and the quality of the solution.In addition, the introduction of a learning parameter, , $\lambda \backslash lambda\lambda$ radically, smoothens the transition process between the exploration-centered behavior at the initial iterations of the algorithm and exploitation centered-improvement at later iterations [48]. This adaptive process improves on the rate of convergence as well as prevents monitoring prematureness. Significantly, the actual contemplation of virtual machine (VM) capacity constraints will make task deployment as practical, as it is balanced to mitigate the possibility of overwhelming VMs and upholding system stability.The implications of the findings cut across the cloud service providers and the end users. To the providers, "HybridPSOGWO" algorithm results in increased utilization of resources through minimizing VM overload and underutilization thanks to their better allocation scheme. To the users, the algorithm will help to complete the tasks faster due to minimization of the makespan and, in this way, increase the perceived Quality of Service (QoS).Moreover, the algorithm proves good scalability, showing a steady performance with an increase in the numbers of tasks [49]. Such an ability renders it adaptable to large and heterogeneous clouds. Therefore, "HybridPSOGWO" is a potential framework in scheduling real-time cloud applications, in which diversity of workloads, and arrival of dynamic tasks are natural issues.Although it has proven to be effective, the "HybridPSOGWO" method is not flawless [50]. To begin with, the hybridization process comes with an extra computational burden over and above the unsophisticated heuristic methodologies. Nevertheless, this trade-off can be deemed as an acceptable one since the significant gains in scheduling performance have been obtained. The other drawback is how sensitive the algorithm is to starting parameter settings; this requires careful tuning of parameters to achieve optimal results.In addition, the current implementation is only capable of

independent task scheduling and fails to consider inter-task dependencies. The framework could be extended to facilitate workflow-based scheduling or other dependent task situations, in so doing, improving its application and increasing its general applicability to more complicated cloud computing situations.

5. Conclusion

In the given study, the researchers offer HybridPSOGWO, a novel hybrid algorithm of the metaheuristic type, that is tailor-made to the task scheduling problem within cloud environments. The algorithm combines Particle Swarm Optimization (PSO) and Grey Wolf Optimizer (GWO) and takes advantage of both algorithms. HybridPSOGWO has the advantages of high scheduling efficiency and robustness and reliability far exceeding all the current scheduling methods; this is through exploiting the fine-grained local exploitation potentialities of PSO and the global exploration potentialities of GWO. The combination has the capacity to allow the algorithm to provide proper task assignments even in dynamic and large-scale cloud computing environments."HybridPSOGWO" The performance was validated by a comprehensive array of assessments within the CloudSim Plus simulation framework and the Google Borg Traces dataset. The results corroborated that the fundamental performance parameters, including makespan, throughput, and load balancing, were significantly improved, hence validating the scheme's practical usefulness. Future work will focus on applying the framework to workflow-based, energy-efficient task scheduling with the goal of improving energy efficiency of cloud-scale deployments. The other interesting direction is the creation of an adaptive version that may work under hazardous conditions of unceasing arrivals and departures of the tasks. Also, the implementation of machine learning models that generate predictions about the features of tasks, and consequently conclusively set the scheduling rules, could bring the predictive intelligence into the scheduling procedure by making it even more flexible and versatile.

REFERENCES

[1] R. Bellman, *Mathematical Optimization Techniques*. Berkeley, CA, USA: Univ. of California Press, 1963.

[2] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19–26, 1989.

[3] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.

[4] T. Bartz-Beielstein, J. Branke, J. Mehnen, and O. Mersmann, "Evolutionary algorithms," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 3, pp. 178–195, 2014.

[5] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: A gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[6] A. Chakraborty and A. K. Kar, "Swarm intelligence: A review of algorithms," in *Nature-Inspired Computing and Optimization: Theory and Applications*, 2017, pp. 475–494.

[7] S. Lipsa, R. K. Dash, N. Ivkovic, and K. Cengiz, "Task scheduling in cloud computing: A priority-based heuristic approach," *IEEE Access*, vol. 11, pp. 27,111–27,126, 2023.

[8] N. Devi, S. Dalal, K. Solanki, S. Dalal, U. K. Lilhore, S. Simaiya, and N. Nuristani, "A systematic literature review for load balancing and task scheduling techniques in cloud computing," *Artificial Intelligence Review*, vol. 57, no. 10, p. 276, 2024.

[9] X. Huang, M. Xie, D. An, S. Su, and Z. Zhang, "Task scheduling in cloud computing based on grey wolf optimization with a new encoding mechanism," *Parallel Computing*, vol. 122, p. 103111, 2024.

[10] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[11] N. Gupta and S. P. Garg, "Improved workflow scheduling using grey wolf optimization in cloud environment," *Int. J. Appl. Eng. Res.*, vol. 12, pp. 8643–8650, 2017.

[12] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. ICNN'95—Int. Conf. Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[13] X. Huang, C. Li, H. Chen, and D. An, "Task scheduling in cloud computing using particle swarm optimization with time-varying inertia weight strategies," *Cluster Computing*, vol. 23, no. 2, pp. 1137–1147, 2020.
[14] Y. Wang and X. Zuo, "An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 5, pp. 1079–1094, 2021.
[15] Q.-Z. Xiao, J. Zhong, L. Feng, L. Luo, and J. Lv, "A cooperative coevolution hyper-heuristic framework for workflow scheduling problem," *IEEE Trans. Services Comput.*, vol. PP, pp. 1–1, Jun. 2019.
[16] Z. Tong, H. Chen, X. Deng, K. Li, and K. Li, "A scheduling scheme in the cloud computing environment using deep Q-learning," *Information Sciences*, vol. 512, pp. 1170–1191, 2020.
[17] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Computing and Applications*, vol. 33, pp. 13,075–13,088, 2021.
[18] M. Ghahari-Bidgoli, M. Ghobaei-Arani, and A. Sharif, "An efficient task offloading and auto-scaling approach for IoT applications in edge computing environment," *Computing*, vol. 107, no. 5, pp. 1–44, 2025.
[19] A. B. Kathole, V. K. Singh, A. Goyal, S. Kant, A. S. Savyanavar, S. A. Ubale, P. Jain, and M. T. Islam, "Novel load balancing mechanism for cloud networks using dilated and attention-based federated learning with coati optimization," *Scientific Reports*, vol. 15, no. 1, p. 15268, 2025.
[20] P. Pirozmand, H. Jalalinejad, A. A. R. Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *J. Ambient Intell. Humanized Comput.*, vol. 14, no. 4, pp. 4313–4327, 2023.
[21] M. S. A. Khan and R. Santhosh, "Task scheduling in cloud computing using hybrid optimization algorithm," *Soft Computing*, vol. 26, no. 23, pp. 13,069–13,079, 2022.
[22] N. Mansouri, B. M. H. Zade, and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Comput. Ind. Eng.*, vol. 130, pp. 597–633, 2019.
[23] M. Agarwal and G. M. S. Srivastava, "An improved PSO algorithm for cloud computing systems," *Int. J. Emerg. Technol. Eng. Res.*, vol. 6, no. 3, pp. 14–18, 2018.
[24] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Computing*, vol. 26, no. 5, pp. 2479–2488, 2023.
[25] D. Ramesh, S. S. Kolla, D. Naik, and R. Narvaneni, "HGWO-MultiQoS: A hybrid grey wolf optimization approach for QoS-constrained workflow scheduling in IaaS clouds," *Simulation Modelling Practice and Theory*, p. 103127, 2025.
[26] J. Aminu, S. Kamarudin, R. Latip, B. U. Kangiwa, Z. M. Hanafi, and A. Liman, "An enhanced grey wolf optimization algorithm for efficient task scheduling in mobile edge computing," *Int. J. Comput. Appl.*, vol. 186, no. 56, pp. 39–44, 2024.
[27] B. Sowjanya and P. Srinivas, "Cloud computing environment: Review on task scheduling algorithms," *Int. J. Eng. Sci. Adv. Technol.*, vol. 18, no. 10, pp. 93–98, 2018.
[28] K. Lv and T. Huang, "Construction of cloud computing task scheduling model based on simulated annealing hybrid algorithm," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 5, pp. 75–82, 2024.
[29] Y. Zhang, Z. Wei, J. Zhang, Z. Zhang, and Y. Shi, "Reinforcement learning-based comprehensive learning grey wolf optimizer for feature selection," *Applied Soft Computing*, vol. 147, p. 110028, 2024.
[30] D. Mwiti and E. Gitonga, "Google 2019 Cluster sample." [Online]. Available: https://www.kaggle.com/datasets/derrickmwiti/google-2019-cluster-sample. Accessed: May 19, 2025.
[31] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Gener. Comput. Syst.*, vol. 25, no. 6, pp. 599–616, 2009.
[32] M. Aazam and E.-N. Huh, "Cloud broker service-oriented resource management model," in *Proc. IEEE 16th Int. Conf. Netw.-Based Inf. Syst.*, 2013, pp. 53–59.
[33] J. Kołodziej and F. Xhafa, "Efficient heuristic-based genetic algorithm for scheduling tasks in computational grids," *Future Gener. Comput. Syst.*, vol. 27, no. 6, pp. 884–893, 2011.
[34] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2018.
[35] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.
[36] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.

[37] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.
[38] M. D. Abualigah, Y. T. H. Abawajy, A. S. Al-Qaness, M. Diabat, and S. Mirjalili, "A comprehensive review of metaheuristics for solving routing problems in wireless sensor networks," *J. Netw. Comput. Appl.*, vol. 174, p. 102890, 2021.
[39] M. Kalra and S. Singh, "A review of metaheuristic scheduling techniques in cloud computing," *Egypt. Inform. J.*, vol. 16, no. 3, pp. 275–295, 2015.
[40] L. Wang, J. Tao, and M. Kunze, "Scientific cloud computing: Early definition and experience," in *Proc. 10th IEEE Int. Conf. High Perform. Comput. Commun.*, 2008, pp. 825–830.
[41] J. Kołodziej and F. Xhafa, "Efficient heuristic-based genetic algorithm for scheduling tasks in computational grids," *Future Gener. Comput. Syst.*, vol. 27, no. 6, pp. 884–893, 2011.
[42] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2–3, pp. 243–278, 2005.
[43] F. Xhafa, J. Kołodziej, and A. Abraham, "A survey of nature-inspired metaheuristics for resource allocation and scheduling in cloud computing," *J. Supercomput.*, vol. 74, pp. 5041–5092, 2018.
[44] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization: An overview," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, 2007.
[45] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.
[46] P. Venkatesan and P. Velusamy, "A grey wolf-based task scheduling algorithm for cloud computing," *Cluster Comput.*, vol. 22, no. 6, pp. 14711–14717, 2019.
[47] S. Pandey and L. Wu, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments," in *Proc. Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 400–407.
[48] S. Kumar and A. Verma, "Task scheduling in cloud computing using hybrid PSO heuristic," *Procedia Comput. Sci.*, vol. 115, pp. 754–761, 2017.
[49] M. Elghamrawy and R. Bahgat, "A hybrid PSO-GWO load balancing algorithm in cloud computing environment," *Int. J. Comput. Appl.*, vol. 177, no. 26, pp. 30–35, 2020.
[50] A. Sharma, M. Sharma, and A. Rajput, "Hybrid metaheuristic algorithm for task scheduling in cloud computing," *J. King Saud Univ. – Comput. Inf. Sci.*, in press, 2022.