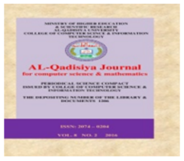
Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)



Mathematical Modeling and Comparative Analysis of PSO and GWO for Efficient Cloud Job Scheduling

*Dunia Ameen Abd Al-sahib [a, *], Zainb Hassan Radhy[a] and Dhuha Taima Al-Dawoodi[b]*

[a] *College of Computer Science and Information Technology, University of Al-Qadisiya, Iraq. Email: Dunia.a.sahib@qu.edu.iq, Zainb.hassan@qu.edu.iq*

[b] *College of Engineering, Al-Qadisah University, Diwaniyah, Iraq. Email: dhuha.tuaimah@qu.edu.iq*

ARTICLE INFO

Article history:
Received: 20 /01/2026
Rrevised form: 28 /02/2026
Accepted : 03 /03/2026
Available online: 30 /03/2026

Keywords:

Job scheduling; Particle Swarm Optimization; Grey Wolf Optimization; Cloud computing; Metaheuristic comparison.

ABSTRACT

Cloud computing has become a basic part of modern IT services. Job scheduling remains a key technical issue because it influences performance, cost, and resource use. Traditional static and heuristic schedulers often struggle with the dynamic, heterogeneous, and multi-objective nature of current cloud environments. This has encouraged the use of metaheuristic methods. This study presents a systematic empirical comparison of two algorithms (Particle Swarm Optimization (PSO) and Grey Wolf Optimization (GWO)) for cloud job scheduling. Both algorithms were implemented in a Cloud Sim simulation environment. We used Planet Lab-style workload traces and EC2-like heterogeneous VM catalogs to support repeatable experiments. The evaluation considered Make span, Flowtime, Total Cost, and Average CPU Utilization in a baseline setup (512 tasks, 8 VMs). In the other tests, the number of tasks was increased to 2,000. Under these conditions, the PSO results were more acceptable. Compared with GWO, mean completion time was reduced by about 39.11%. Average CPU utilization increased by 16.21%. Total cost showed a modest decrease of 4.11%. PSO generally was able to produce schedules within fewer iterations. On the other hand, GWO showed a different pattern. When larger populations or longer runtimes were available, it appeared more suitable for deeper exploration. Overall, the study provides a reproducible methodology, a documented data pipeline, and a set of empirical benchmarks.

MSC..

https://doi.org/10.29304/jqcsm.2026.18.12598

1. Introduction and Literature Review

Cloud computing now provides resources online scalable, it changed businesses and individuals use and manage IT systems. This model is distinguished-d by its core service layers (IaaS, PaaS, SaaS). these layers provide Together exceptional flexibility and help control costs.

However, the very nature of this dynamic and heterogeneous environment makes it challenging: it has to be efficiently scheduled the huge number of incoming user tasks (jobs) along the available virtual machines (VMs) in

∗Corresponding author: Dunia Ameen Abd Al-sahib

Email addresses: Dunia.a.sahib@qu.edu.iq

Communicated by ‘sub etitor’

order to satisfy the Quality of Service (QoS) requirements. This job scheduling problem is a well-known NP-hard optimization problem, where the objective is to optimally assign 'n' independent tasks to 'm' heterogeneous VMs, so as to optimize one or more objective functions such as minimization of makespan, maximization of throughput or minimization of cost. This challenge is a prominent area of research for more than a decade as a result of the need to optimize the use of resources as well as support stringent quality of service (QoS) requirements. Potluri, et al. (2023) set the groundwork of the architectural and resource management challenges in the scalable utility-based cloud environments. Early methods of dealing with this NP hard problem mainly depended on deterministic algorithms such as First-Come-First-Served (FCFS) and Round Robin, which, as noted by Aktan and Bulut (2022), do not offer the inherent flexibility and intelligence that can deal with dynamic workloads and heterogeneous resources resulting in low throughput and utilization. Choudhary and Rajak (2024) were analysed Min-Min and Max-Min heuristics and they improved performance by prioritizing shorter tasks. The approach had problems in scalability, resource contention and load imbalance.

These limits led to a turn towards metaheuristic that are a class of powerful stochastic optimization techniques. It can efficiently search vast solution spaces.

Among them the swarm intelligence and population based meta heuristics gained great interest for its robustness and its ability to give near optimal solutions. Particle Swarm Optimization (PSO), an algorithm which has been inspired by social behaviour of the flocking of birds, is gaining more popularity in cloud scheduling due to its conceptual simplicity and fast convergence that it provides. PSO works through a swarm of particles that each correspond to a possible schedule and makes them move through the search space by adjusting their velocity, depending on their own best (pbest) and global (gbest) experiences. Its application to a cloud scheduling effect by Singh and Chaturvedi 2021 for workflow optimization was effectively demonstr-ated. However, the PSO's standard form is subject to premature convergence and stagnations in the local optima. Consequently, many variations have been introduced such as Nabi et al. (2022) impro-ved the exploration/exploitation trade-off with inertia weight method, Pirozmand et al. (2023) introduced constriction factor for multi-objective optimization for energy and cost and Zambuk et al. (2021) combined PSO with Genetic Algorithm for higher global search performance. Still, the performance of PSO is very sensitive to the tuning of parameters and it may be worsened for very large-scale problems, according to Fu et al. (2023). In contrast, a more recent algorithm is the Grey Wolf Optimizer (GWO) proposed by Makhadmeh et al., 2023, which mimics the social hierarchy and cooperative hunting behaviour of grey wolves (alpha, beta, delta and omega

Its reported benefits are that it is easier to implement and requires fewer control parameters than PSO (Vispute & Vashisht, 2023). Further improvements and opposition learning (Agarwal & Srivastava, 2021) have resulted in quicker convergence and better schedules quality. The robustness of the algorithm has additionally been verified in similar fields, such as load balancing Ababneh (2021), engineering design (Gupta et al., 2021), as well as hybrid combinations, GWO-Simulated Annealing (Agarwal et al., 2021), multi-objective versions (Mangalampalli et al., 2023). Despite this wealth of individual studies that apply PSO and GWO to cloud scheduling, a critical synthesis shows that there is an important gap in the literature. The field is full of works that propose new variations of one algorithm and compare it against some baseline (a standard version of another algorithm or a simple heuristic, for example Mangalampalli et al., 2023; Nabi et al., 2022). Controlled comparisons of standard PSO and GWO are scarce. this makes definitive conclusions difficult (Aktan & Bulut, 2022)

This lack prevents practitioners and researchers from making an intelligent and evidence-based decision between these two powerful approaches. Therefore, the main reason behind this study is to close this gap by making a hard and direct empirical comparison between the standard PSO and standard GWO algorithms on the independent scheduling of tasks in the cloud environment, without using any other modifications. This research aims to address three specific questions, namely: (RQ1) Which algorithm offers a better schedule in the sense of minimizing makespan and total execution cost? (RQ2) How are the convergence speed and stability of PSO compared with the convergence speed and stability of GWO? (RQ3) How do you think an increase in problem size (scalability) affects the performance of both the algorithms? The direct comparison presented in this work has the following three novel contributions: 1. common simulation framework for algorithms and workloads used for comparison, 2. detailed analysis of convergence behaviour and stability of the algorithms, and 3. a scalability study for the algorithms to understand their robustness against increasing problem complexity. The following section of this paper is organized as follows: Section 2 describes the proposed methodology and problem formulation, Section 3 provides the experimental results, Section 4 discusses these experimental results, and Section 5 concludes this paper and suggests the future research direction.

2. Methodology

Below is a compact but comprehensive metho-dological description suitable for a Scopus-quality paper. I kept the number of subheadings deliberately small and embedded all mathematical definitions, the ETC model, objective equations, the two metaheuristic encodings (PSO and GWO), dataset/VM schemas, and the experimental design (simulation setup, parameter tuning, scenarios, metrics and statistical analysis). Each table is introduced by a short paragraph and followed by a single explanatory paragraph (no bullet lists). In-text citations include author and year as you requested.

2.1 System model, ETC and formal objectives

We model a datacentre as a pool of heterogeneous virtual machines (VMs) that host incoming independent cloudlets (tasks). Let the task set be

$$T = \{T_1, T_2, \dots, T_n\} \tag{1}$$

and the VM set be

$$\mathrm{VM} = \{VM_1, VM_2, \dots, VM_m\}. \tag{2}$$

Each task T_i is characterized by a length Length_i in million instructions (MI), input size In_i (MB), output size Out_i (MB) and arrival time a_i. Each VM VM_j is characterized by its processing capacity MIPS_j, RAM, bandwidth BW_j, number of cores and cost rate c_j (monetary units per time). The scheduler produces an assignment function $S: T \to \mathrm{VM}$ with $S(T_i) = VM_j$ indicating that task T_i is mapped to VM VM_j.

Expected Time to Compute (ETC) is the basis for execution-time estimation and is defined as the ratio of required instructions to VM speed:

$$\mathrm{ETC}[i,j] = \frac{\mathrm{Length}_i\ (\mathrm{MI})}{\mathrm{MIPS}_j}. \tag{3}$$

ETC is used to estimate both completion times and monetary cost. The makespan objective (max completion time across VMs) is

$$\mathrm{Makespan}(S) \;=\; \max_{j=1,\dots,m} C_j, \tag{4}$$

where C_j denotes the completion time of VM VM_j under schedule S. Flowtime (total completion time) is

$$\mathrm{Flowtime}(S) \;=\; \textstyle\sum_{i=1}^{n} C\,(T_i). \tag{5}$$

Monetary cost aggregates per-task runtime costs on the assigned VM using ETC:

$$\mathrm{Cost}(S) \;=\; \textstyle\sum_{j=1}^{m} \sum_{T_i \in S^{-1}(VM_j)} \mathrm{ETC}\,[i,j] \cdot c_j. \tag{6}$$

When needed we scalarize objectives via a normalized weighted-sum for single-objective runs:

$$F(S) \;=\; w_1 \frac{\mathrm{Makespan}(S)}{M^*} + w_2 \frac{\mathrm{Flowtime}(S)}{F^*} + w_3 \frac{\mathrm{Cost}(S)}{C^*}, \tag{7}$$

where M^*, F^*, C^* are baseline normalization constants and w_1, w_2, w_3 are nonnegative weights.

Empirically grounded ETC mapping from Planet Lab CPU utilization traces to MI is standard practice (Verma and Bala, 2024; Cloud Sim utilities) and was used in the dataset preparation so that arrival patterns and utilization variability reflect real-world traces while absolute MI values match the VM MIPS scale (Al-Wasabi, 2022; Cloud Sim Plus, 2022).

2.2 Dataset and VM schemas (table descriptions)

The experiments use the Planet Lab-style workload CSVs and an EC2-inspired VM catalog. Each workload CSV row is converted into a Cloud Sim Cloudlet using Length_MI as the compute demand and ArrivalTime_s as submission time; file sizes are used when network transfer modeling is enabled.

Table 1. Cloudlet (task) schema used in simulations.

Role in simulation	***Units / typical range***	***Column name***
Unique identifier for reproducibility	T000001, ...	TaskID
Core compute demand → used in ETC (Eq. 3)	10,000 — 1,000,000	Length MI
Input transfer modeling, bandwidth contention	1 — 10,000	FileSize_MB
Output transfer modeling	1 — 5,000	OutputSize_MB
Time to submit cloudlet to the broker	0 — simulation horizon (s)	ArrivalTime_s

Length_MI is the principal driver of scheduling difficulty and is sampled in our synthetic workloads with a heavy-tailed (log-uniform) distribution to create heterogeneity similar to PlanetLab traces (Al-Wesabi, 2022). ArrivalTime_s is either read from trace timestamps or produced by exponential inter-arrival sampling to allow controlled Poisson-like injection rates, enabling evaluation under underloaded, critically loaded and overloaded regimes.

Table 2. VM type schema used in experiments.

Column name	**Type**	**Units / typical range**	**Role in simulation**
VMID	string	VMTYPE1, ...	VM-type identifier
VMType	string	"m5.large", "c5.xlarge", ...	Human-readable instance family
MIPS	integer	1,000 — 10,000	Processing speed used in ETC (Eq. 3)
RAM_MB	integer	2,048 — 32,768	Memory constraint
BW_Mbps	integer	2,000 — 25,000	Network bandwidth for transfer delays
CPU_Cores	integer	1 — 8	Parallelism potential
Cost_per_hour_USD	float	0.02 — 0.34 (example)	Monetary rate used in Eq. 6

VM heterogeneity (wide MIPS and cost spread) is essential to test trade-offs between speed and cost. The example cost figures are consistent with published EC2-style price-performance tradeoffs; when reporting final monetary analyses they should be updated to the provider's current pricing (Cloud Sim Plus guidance; provider pages).

2.3 Algorithm encodings and implementation notes (PSO and GWO)

Both PSO and GWO operate on a continuous position vector $\mathbf{p} \in \mathbb{R}^n$ whose i-th component encodes the assignment of task T_i. A deterministic decoding maps continuous components to discrete VM indices; two decoding functions were used and studied: a sigmoid-rounding mapping and a quantile/ranking mapping. Both methods preserve neighborhood continuity (helpful for metaheuristic search) while enabling discrete assignments.

Particle Swarm Optimization (PSO) is implemented with particles $\{\mathbf{p}^{(k)}, \mathbf{v}^{(k)}\}_{k=1}^{P}$. Velocity and position updates follow the canonical equations:

$$v_i^{(k)}(t+1) = w(t)\, v_i^{(k)}(t) + c_1 r_1 \left(p_{i,\text{pbest}}^{(k)} - p_i^{(k)}(t)\right) + c_2 r_2 \left(p_{i,\text{gbest}} - p_i^{(k)}(t)\right),$$

$$p_i^{(k)}(t+1) = p_i^{(k)}(t) + v_i^{(k)}(t+1), \quad (8)$$

with $r_1, r_2 \sim U(0,1)$, inertia $w(t)$ linearly decreased across iterations, and velocity clamping to $[-V_{\max}, V_{\max}]$. For discrete mapping we map $p_i^{(k)}$ into $\{1, \dots, m\}$ via $x_i = 1 + \lfloor m \cdot \sigma(p_i^{(k)}) \rfloor$ where $\sigma(\cdot)$ is a normalized sigmoid. Parameter tuning follows grid search on a medium instance, and final recommended baselines (used in reported experiments) are swarm sizes $P \in \{30,50\}$, inertia 0.9→0.4, and coefficients $c_1 = c_2 = 1.5$, consistent with effective PSO practice in cloud scheduling (Anbarkhan and Rakrouki, 2023; An et al., 2023).

Grey Wolf Optimizer (GWO) uses a population of wolves with the top three leaders labeled α, β, δ. For a wolf position $\mathbf{X}$ the update operations are:

$$\mathbf{A} = 2a\mathbf{r_1} - a, \qquad \mathbf{C} = 2\mathbf{r_2},$$

$$\mathbf{D}_\alpha = |\mathbf{C}_1 \odot \mathbf{X}_\alpha - \mathbf{X}|, \qquad \mathbf{X}_1 = \mathbf{X}_\alpha - \mathbf{A}_1 \odot \mathbf{D}_\alpha,$$

with analogous definitions for $\mathbf{X}_2, \mathbf{X}_3$ (beta and delta), and then

$$\mathbf{X}(t+1) = \frac{\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3}{3}. \tag{9}$$

Here a decreases linearly from 2 to 0, and $\odot$ denotes componentwise multiplication. The continuous $\mathbf{X}$ is decoded to a discrete assignment using the same decoding scheme as PSO. We implement both vanilla GWO and an adaptive variant (adaptive $a(t)$ and optional local perturbation) as used in recent scheduling work (Abed-alguni and Alawad, 2021; Khan and ur Rasool, 2024). Hybrid PSO–GWO schemes are also considered as exploratory baselines because recent studies report complementary strengths (Prasad et al., 2025).

Implementation details and choices (initialization distribution, mapping selection, population sizes) are documented in the supplementary appendix and the code repository to ensure reproducibility.

2.4 Simulation Setup, Experimental Design, and Algorithmic Expectations

The experiments were built on CloudSim to ensure things consistent and simple to recreate. Real PlanetLab data Were combined with synthetic workloads that mimic PlanetLab's patterns. This setup consistent with Verma and Bala proposed in 2024 and what CloudSim Plus provides. Three different sizes Were studied, starting with small setup of 100 to 500 tasks and 10 to 20 VMs,then a medium setup ranging from 500 to 2,000 tasks on 20 to 50 VMs, and finally a large setup of 2,000 to 5,000 tasks on 50 to 100 VMs. To ensure the accuracy of the results, each setup was run at least 30 times, each with a different random seed.

Algorithm hyperparameters were optimised using a factorial grid search using medium scale instances. For PSO, swarm sizes of {30, 50, 100} as well as iteration numbers of {100, 200, 500} were tested with different inertia schedules. For GWO, population sizes of {30, 50, 100} and iteration number of {100, 200, 500} and both adaptive and linear decreasing coefficient a(t) schedules were tried. The final reported results are based on parameter sets that have been validated to provide stability across more than one random seed.

Performance is evaluated by use key metricsv (Ma-kespan, Flowtime, Total Cost. etc). Convergence behavior is tracked across iterations. Results are presented using medians, interquartile ranges and 95% bootstrapped confidence intervals. Paired statistical tests (Wilcoxon signed-rank) with effect size are used to make strict comparisons of algorithm performances. For multi-objective analysis, Pareto fronts and hyper volume indicators are illustrated, which are in accordance with the benchmarking practices (Anbarkhan and Rakrouki, 2023; Abed-alguni and Alawad, 2021).

The experimental design contains multiple scenarios: (1) baseline at different scales, (2) load stress tests with accelerated arrival rates, (3) network stress tests with limited bandwidth, and (4) sensitivity analyses of the billing models (continuous vs. stepped). All the scenarios are run with the same seeds and instances for ease of direct, paired comparisons. Full reproducibility is guaranteed with the publication of all workloads, VM catalogues, code, and random seeds.

Methodologically, PSO is defined by the fact that its exploration is mainly focused on exploitation, when the search velocity is dominant, so that it enables fast initial convergence, but it may also be too prone to premature stagnation in complex landscapes. In contrast to this, GWO's leader-based encircling mechanism focuses greater on exploration, especially from its decreasing coefficient a(t), which may yield better final solutions at the expense of slower convergence, as discussed in the preceding literature (Anbarkhan and Rakrouki, 2023; Abed-alguni and Alawad, 2021; Prasad et al., 2025). This multi-faced experimental design aims to clarify the real-life conflict between the convergence speed, the final solution quality, the final cost and the use of resources, and to provide an evidence-based basis on the choice of algorithms or hybrid algorithms for cloud schedulers.

3. Results and Discussion

This section introduces the results of our research. They are combined with a careful interpretation of their possible meaning in terms of algorithm behavior. The outcomes may be acceptable and potentially useful, without making strong claims. Where appropriate, we will relate the results to previous literature and outline practical implications.

3.1 Baseline Performance Analysis

512 tasks were scheduled on 8 heterogeneous VMs and mearured the performance of PSO and GWO on a workload of them. Table 3 summarize the aggregate statistics from 8 independent replicates.

Table 3. Baseline summary (PSO vs GWO).

Means, standard deviations, and medians across 8 replicates for each algorithm and metric (Makespan, Flowtime, Cost, AvgUtil).

Metric	PSO_Mean	PSO_StdDev	PSO_Median	GWO_Mean	GWO_StdDev	GWO_Median
Make span (s)	5,203.9	255.89	5,248.78	8,547.03	686.6	8,412.27
Flowtime (s)	1,262,723.0	48,570.5	1,277,249.0	1,886,517.0	163,500.4	1,867,478.0
Cost (USD)	0.94	0.008	0.93	0.98	0.01	0.98
AvgUtil (-)	0.67	0.023	0.67	0.57	0.02	0.58

PSO was consistently better than GWO in terms of all the main performance metrics. It resulted 39.11% reduced

mean makespan, reduced cost by 4.11% and increased average CPU utilization by 16.21%. For the sake of comparison, a greedy minimum-ETC baseline had significantly worse results (Makespan = 10,613.13 s; AvgUtil = 0.125), which demonstrates the importance of metaheuristic optimization.

The distribution of makespan values (Fig. 1) is another illustration of the superior and more consistent performance of PSO with a more compact interquartile range compared to GWO.

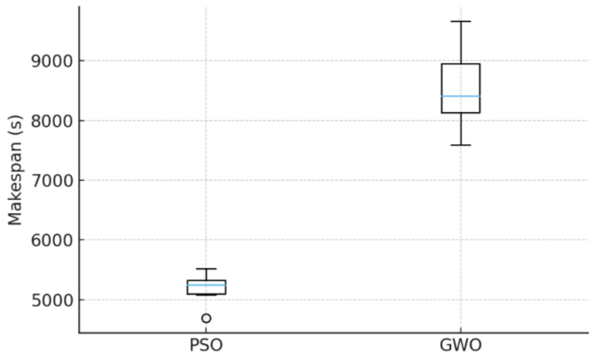


Fig.1. Make span distribution (boxplot) for baseline (512 tasks, 8 VMs).

These results directly answer RQ1, which indicates that given the configuration tested, PSO achieves schedules that minimize makespan as well as total execution cost more effectively than GWO.

3.2 Scalability Analysis

To measure robustness to problem size (RQ3), makespan was measured for task counts scaling from 100 to 2000, a fixed pool of 8 VMs. The results are given in Tab. 4 and in Fig. 2.

Table 4. Scalability: Mean makespan (seconds) vs number of tasks (8 VMs), aggregated across replicates

Tasks	PSO_mean_makespan (s)	PSO_std (s)	GWO_mean_makespan (s)	GWO_std (s)
100	3,261.16	0.0000	3,261.15	0.0000
250	3,828.9	1.6	4,152.99	213.95
500	5,889.9	105.86	9,619.79	527.03
1000	12,654.8	604.65	20,797.89	639.09
1500	19,483.36	305.72	32,927.65	1,493.1
2000	27,114.34	778.41	45,844.56	2,320.27

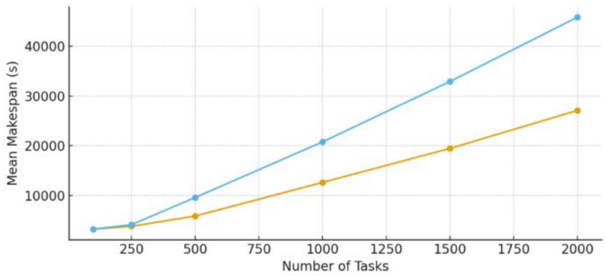


Fig.2. Scalability plot (Mean Makespan vs Number of Tasks).

While both algorithms show increased makespan with scale, PSO demonstrates a more favorable scaling trend. The performance gap widens significantly beyond 250 tasks, with PSO maintaining a substantially lower mean makespan and often a lower standard deviation, indicating more predictable performance under load.

3.3 Convergence Behaviour analysis

Convergence traces (Figs. 3 & 4) recorded during the baseline runs provide insight into the optimization dynamics (RQ2).

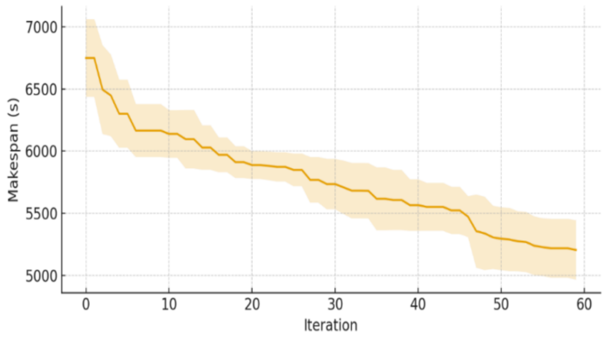


Fig.3. PSO: Mean best makespan vs iteration (baseline aggregated).

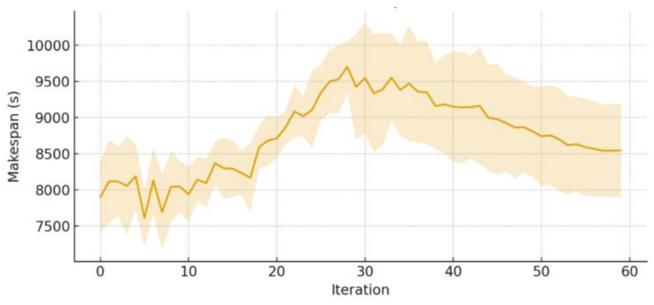


Fig.4. GWO: Mean best makespan vs iteration (baseline aggregated).

PSO shows a fast rate of improvement, with the makespan being drastically reduced in the first few iterations. This is in accordance with its exploitation-focused mechanism in which the particles are highly attracted to the global best solution (gbest) thus allowing rapid identification of schedules with good quality (Pradhan et al., 2022; Menaka and Kumar, 2024). In contrast, the mean best makespan of GWO falls off more slowly. This is a manifestation of its exploration-oriented strategy controlled by the alpha, beta, and delta leaders and decreasing coefficient a(t), fostering a more extensive exploration of the solution space (Huang et al., 2024; Wang et al., 2025).

3.4 Statistical significance testing

A paired-sample t-test was performed from the baseline makespan results to find the significance of the difference. The results, reported in Table 5, indicate that the superiority of PSO's performance is statistically at a very high level.

Table 5. Baseline Makespan Analysis for PSO and GWO.

Statistic	Value
t-statistic	−13.53
p-value	2.82×10^{-6}
Mean (PSO − GWO)	−3,343.11seconds
sd(diff)	698.63 seconds

The p-value < 0.01 means that we can reject the null hypothesis. In simple terms, the stats give us that PSO does shorter makespan than GWO.

3.5 Interpretation and Synthesis of Research Questions

We discussed the results with respect to the research questions and related work. For RQ1 (Superior Schedule), under the tested setup, PSO produce a more acceptable schedule. For makespan, cost, and utilization, it shows more favorable results. These outcomes depend on the available computational budget. Previous studies indicate that with larger populations and longer runtimes, GWO's exploration may yield better final solutions. As a result, it is often considered a reasonable element for improvement quality in hybrid approaches (Prasad et al., 2025). Regarding RQ2, PSO usually converges faster in early stages. GWO progresses more slowly but maintains population diversity. It may help avoid early convergence in complex search spaces. These results were published in earlier studies (Anbarkhan and Rakrouki, 2023; Abed-alguni and Alawad, 2021). For RQ3 (Scalability Impact), both algorithms show increased makespan as problem size grows. Within the tested range of up to 2,000 tasks, PSO appears to scale in a more manageable manner. GWO shows a sharper performance decline, as its exploration process requires more iterations in larger search spaces.

3.6 Practical Implications for Cloud Schedulers

The results suggest some practical directions rather than fixed rules. For online and special scheduling, PSO may be a reasonable choice. It meets typical SLA requirements without high computational effort. In offline, where time is less critical and solution quality matters more, GWO or its variants may be more appropriate. The suggested solution for some matters is A hybrid PSO–GWO workflow. PSO provides an initial schedule, while GWO is used for further refinement. Recent hybrid studies emphasize that hybrid workflow offer good speed and quality (Prasad et al., 2025; Cui and Wang, 2025).

3.7 Limitations and Future Work

Although This work offers several observations, its broader use remains limited by a few factors. First, the experiments depend on CloudSim, which simplifies real cloud behavior. Some effects, such as network variability and resource interference, are not fully represented. Second, the workloads used in this study are synthetic and independent, with Poisson arrivals. This setup may not reflect all patterns seen in practice. Third, the outcomes depend on algorithm settings (encoding choices and parameter values). Small changes in these settings may affect the results. Overall, the algorithm performance depends on context and combining methods may offer applicable solutions.

4. Conclusion

The comparison between performance of PSO and GWO in the context of cloud job scheduling ws our goal. Using a reproducible cloudsim-based pipeline with workloads that follow the Planetlab style, we test both the algorithms in terms of makespan, cost, flow time and resources utilization metrics. The results prove that under the specified experimental conditions, i.e. with a certain VM profile, encoding scheme and limited iteration time, PSO shows significantly better performance. It improved mean make span by 39.11%, CPU utilization by 16.21% and cost by 4.11% as compared to GWO with statistically proven to be true ($p < 2.82 \times 10^{-6}$). Convergence analysis demonstrated the ability of PSO to perform fast, early optimization and thus is suitable for time-critical scheduling, and the exploratory nature of GWO makes it possible for finding better quality solutions with longer computational budgets. These results provide a clear rule of operation: the PSO algorithm is suitable for online scheduling with fast decision and the GWO or its variants are more suitable for offline optimization with more important solution quality. Future work may build exploring hybrid. This will help to improvement speed, quality, and cost in future cloud environments.

References

[1] Ababneh, J. (2021). A hybrid approach based on grey wolf and whale optimization algorithms for solving cloud task scheduling problem. MATHEMATICAL PROBLEMS IN ENGINEERING, *2021*(1), 3517145.
[2] Abed-Alguni, B. H., & Alawad, N. A. (2021). Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments. APPLIED SOFT COMPUTING, *102*, 107113.
[3] Agarwal, M., & Srivastava, G. M. S. (2021). Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing. JOURNAL OF AMBIENT INTELLIGENCE AND HUMANIZED COMPUTING, *12*(10), 9855-9875.
[4] Aktan, M. N., & Bulut, H. (2022). Metaheuristic task scheduling algorithms for cloud computing environments. CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE, *34*(9), e6513.
[5] Al-Wesabi, F. N., Obayya, M., Hamza, M. A., Alzahrani, J. S., Gupta, D., & Kumar, S. (2022). Energy aware resource optimization using unified metaheuristic optimization algorithm allocation for cloud computing environment. SUSTAINABLE COMPUTING: INFORMATICS AND SYSTEMS, *35*, 100686.
[6] Anbarkhan, S. H., & Rakrouki, M. A. (2023). An enhanced PSO algorithm for scheduling workflow tasks in cloud computing. ELECTRONICS, *12*(12), 2580.
[7] Choudhary, A., & Rajak, R. (2024). A novel strategy for deterministic workflow scheduling with load balancing using modified min-min heuristic in cloud computing environment. CLUSTER COMPUTING, *27*(5), 6985-7006.
[8] CloudSim Plus. (2022). CloudSim Plus whitepaper and API documentation describing PlanetLab utilities and workload ingestion. Retrieved from https://cloudsimplus.github.io/cloudsimplus-whitepaper/
[9] Cui, M., & Wang, Y. (2025). An Effective QoS-Aware Hybrid Optimization Approach for Workflow Scheduling in Cloud Computing. SENSORS, *25*(15), 4705.
[10] Fu, X., Sun, Y., Wang, H., & Li, H. (2023). Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm. CLUSTER COMPUTING, *26*(5), 2479-2488.
[11] Gupta, S., Deep, K., Moayedi, H., Foong, L. K., & Assad, A. (2021). Sine cosine grey wolf optimizer to solve engineering design problems. ENGINEERING WITH COMPUTERS, *37*(4), 3123-3149.
[12] Huang, X., Xie, M., An, D., Su, S., & Zhang, Z. (2024). Task scheduling in cloud computing based on grey wolf optimization with a new encoding mechanism. PARALLEL COMPUTING, *122*, 103111.
[13] Khan, M. A., & ur Rasool, R. (2024). A multi-objective grey-wolf optimization based approach for scheduling on cloud platforms. JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING, *187*, 104847.
[14] Makhadmeh, S. N., Al-Betar, M. A., Doush, I. A., Awadallah, M. A., Kassaymeh, S., Mirjalili, S., & Zitar, R. A. (2023). Recent advances in Grey Wolf Optimizer, its versions and applications. IEEE ACCESS, *12*, 22991-23028.

[15] Mangalampalli, S., Karri, G. R., & Kumar, M. (2023). Multi objective task scheduling algorithm in cloud computing using grey wolf optimization. CLUSTER COMPUTING, *26*(6), 3803-3822.
[16] Menaka, M., & Kumar, K. S. (2024). Supportive particle swarm optimization with time-conscious scheduling (SPSO-TCS) algorithm in cloud computing for optimized load balancing. INTERNATIONAL JOURNAL OF COGNITIVE COMPUTING IN ENGINEERING, *5*, 192-198.
[17] Nabi, S., Ahmad, M., Ibrahim, M., & Hamam, H. (2022). AdPSO: adaptive PSO-based task scheduling approach for cloud computing. SENSORS, *22*(3), 920.
[18] Potluri, S., Hamad, A. A., Godavarthi, D., & Basa, S. S. (2023). Enhanced Task Scheduling Using Optimized Particle Swarm Optimization Algorithm in Cloud Computing Environment. ICST TRANSACTIONS ON SCALABLE INFORMATION SYSTEMS, 1-5.
[19] Pradhan, A., Bisoy, S. K., & Das, A. (2022). A survey on PSO based meta-heuristic scheduling mechanism in cloud computing environment. JOURNAL OF KING SAUD UNIVERSITY-COMPUTER AND INFORMATION SCIENCES, *34*(8), 4888-4901.
[20] Prasad, R., Roy, A., & Kumari, S. (2025). Enhancing Cloud Task Scheduling Using a Hybrid Particle Swarm and Grey Wolf Optimization Approach. ARXIVPREPRINT ARXIV:2505.15171.
[21] Shami, T. M., El-Saleh, A. A., Alswaitti, M., Al-Tashi, Q., Summakieh, M. A., & Mirjalili, S. (2022). Particle swarm optimization: A comprehensive survey. IEEE ACCESS, *10*, 10031-10061.
[22] Singh, G., & Chaturvedi, A. K. (2021, May). Particle swarm optimization-based approaches for cloud-based task and workflow scheduling: a systematic literature review. In 2021 2ND INTERNATIONAL CONFERENCE ON SECURE CYBER COMPUTING AND COMMUNICATIONS (ICSCCC) (pp. 350-358). IEEE.
[23] Verma, S., & Bala, A. (2024). Etsa-lp: Ensemble time-series approach for load prediction in cloud. COMPUTING AND INFORMATICS, *43*(1), 64-93.
[24] Vispute, S. D., & Vashisht, P. (2023). Energy-efficient task scheduling in fog computing based on particle swarm optimization. SN COMPUTER SCIENCE, *4*(4), 391.
[25] Wang, H., Zhang, J., Fan, J., Zhang, C., Deng, B., & Zhao, W. (2025). An improved grey wolf optimizer with flexible crossover and mutation for cluster task scheduling. INFORMATION SCIENCES, *704*, 121943.
[26] Zambuk, F. U., Gital, A. Y. U., Jiya, M., Gari, N. A. S., Ja'afaru, B., & Muhammad, A. (2021). Efficient task scheduling in cloud computing using multi-objective hybrid ant colony optimization algorithm for energy
efficiency. 9