






Paper Type: Original Article

Optimization of Virtual Machine Allocation in Cloud Data Centers Using Hybrid WOA-PSO Algorithm

Saeed Mirpour Marzuni¹ , Ali Ghanbari Sorkhi¹ , Mohammad Gholami¹ 

¹ Department of Electrical and Computer Engineering, University of Science and Technology of Mazandaran, Behshahr, Iran; mirpour@mazust.ac.ir; ali.ghanbari@mazust.ac.ir; gholami@mazust.ac.ir.

Citation:

Received: 22 August 2024

Revised: 14 October 2024

Accepted: 20 December 2024

Mirpour Marzuni, S., Ghanbari Sorkhi, A., & Gholami, M. (2025). Optimization of virtual machine allocation in cloud data centers using hybrid WOA-PSO algorithm. *Transactions on Soft Computing*, 1(2), 71-82.

Abstract

Task scheduling in cloud computing is a key and challenging process that directly impacts overall system performance, resource utilization, and user satisfaction. Despite the notable success of traditional metaheuristic algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithm (GA), these methods often face inherent limitations including premature convergence and insufficient exploration of the solution space. To address these challenges, this paper proposes a novel hybrid metaheuristic framework combining the Whale Optimization Algorithm (WOA) with PSO for efficient task scheduling in cloud environments. The hybrid approach leverages WOA's ability to maintain a dynamic balance between global exploration and local exploitation, inspired by the foraging behavior of humpback whales, while utilizing PSO's fast convergence characteristics to accelerate the search process. Tasks are mapped to virtual machines with the objective of minimizing overall makespan. Extensive experiments conducted within the CloudSim simulation environment demonstrate that the proposed hybrid algorithm significantly outperforms standalone PSO, WOA, and hybrid PSO-GA approaches in terms of convergence speed, solution quality, and load balancing. These results confirm the effectiveness and robustness of the proposed hybrid metaheuristic in navigating the complex and dynamic optimization landscape inherent in cloud computing task scheduling problems.

Keywords: Cloud computing, Task scheduling, Whale optimization algorithm, Particle swarm optimization, Resource allocation, Makespan minimization.

1 | Introduction

In recent years, complex optimization problems across various domains, including engineering, artificial intelligence, and data science, have posed significant challenges. Classical optimization algorithms often struggle when dealing with high-dimensional, nonlinear, and multi-modal problems, frequently encountering issues such as premature convergence, entrapment in local optima, and an inadequate balance between

 Corresponding Author: mirpour@mazust.ac.ir



Licensee System Analytics. This article is an open-access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0>).

exploration and exploitation. Consequently, the development of hybrid algorithms that simultaneously leverage the strengths of multiple optimization techniques has emerged as an active area of research.

Particle Swarm Optimization (PSO) is one of the most widely used metaheuristic methods, inspired by the social behavior of bird flocks. It has gained considerable attention due to its simplicity, fast convergence rate, and strong capability to explore the search space [1]. However, PSO sometimes lacks sufficient precision in exploiting promising regions of the search space. On the other hand, the WOA, inspired by the hunting behavior of humpback whales, offers competitive exploitation performance through unique search mechanisms such as spiral updating and prey encircling strategies [2]. Nevertheless, WOA may exhibit limited exploration efficiency during the initial search stages.

Combining these two algorithms offers a promising solution to overcome their respective limitations. This paper presents a novel hybrid approach based on PSO and WOA, specifically designed to achieve an optimal balance between exploration and exploitation. The proposed method employs PSO mechanisms for efficient initial exploration and leverages WOA's exploitation capabilities to refine the search in promising regions. To evaluate the performance of the proposed hybrid method, a set of standard benchmark functions and realistic optimization problems are employed. The obtained results demonstrate that the hybrid approach outperforms the baseline algorithms and several other hybrid methods in terms of optimization efficiency and solution quality.

2 | Related Work

In recent years, many studies have explored metaheuristic algorithms to improve optimization performance in task scheduling. Akbari, [3], proposed an enhanced Genetic Algorithm (GA) for scheduling tasks on heterogeneous systems, modeling tasks as Directed Acyclic Graphs (DAGs). The algorithm introduces novel operators and a final scheduler, resulting in improved makespan and efficiency. Its convergence and effectiveness were rigorously demonstrated using Markov chain theory. Experimental results indicate that this method outperforms previous algorithms and achieves faster convergence in early generations.

Task scheduling in cloud environments is a critical and NP-hard challenge, prompting extensive research into metaheuristic and hybrid approaches. Among swarm-based algorithms, PSO and WOA have both shown promise yet suffer from individual limitations: PSO often converges prematurely, while WOA may lack fast convergence in high-dimensional spaces [4].

Hybrid PSO–GA frameworks have previously attempted to counter these issues. For instance, authors in [5] integrated GA's mutation and crossover operators into PSO to enhance population diversity and mitigate local optima entrapment, demonstrating a clear performance boost in multimodal benchmark optimization.

Focusing on hybrid WOA–PSO approaches, OWPSO [6] was proposed, a hybrid algorithm that incorporates Opposition-Based Learning (OBL) and PSO mechanisms into WOA's exploration phase and switches dynamically between phases. Experiments using CloudSim on bag-of-tasks workloads showed improvement in makespan and in energy savings compared to standard metaheuristics.

Other hybridizations of WOA include combining WOA with Fuzzy-PSO systems and Enhanced WOA variants. For example, In Zhang and Wang, [7] WOA is Enhanced With Lévy Flights (EWOA), achieving significant improvements in resource utilization, energy consumption, and cost minimization in CloudSim-based experimental studies.

Additionally, Strumberger et al. [8] have successfully merged WOA with algorithms like ABC and DE, yielding measurable gains in convergence and solution quality over standard WOA.

While these hybrid models effectively balance exploration and exploitation, most still rely on static switching mechanisms or focus on single-objective improvements. The proposed WOA–PSO hybrid framework in this work advances the field by dynamically integrating PSO's rapid convergence with WOA's local refinement. Our experiments, conducted with CloudSim, validate its superior performance in makespan reduction, load

balancing, and convergence robustness—clearly outperforming standalone and other hybrid metaheuristics in multi-scenario virtual machine scheduling.

Chen et al. [9], applied the WOA to cloud task scheduling using a multi-objective optimization model aimed at improving system performance with limited computing resources. They further introduced an enhanced variant, the Improved WOA for Cloud task scheduling IWC, to boost the original algorithm's search capability. Their simulation results demonstrated that IWC achieved faster convergence and higher solution accuracy compared to traditional metaheuristics, while also improving resource utilization across both small- and large-scale task scenarios.

An Improved Whale Optimization Algorithm (IWOA) was introduced in Han et al. [10] to solve limited buffer scheduling in flexible flow shops with setup times. By integrating Lévy flight, OBL, and simulated annealing, the algorithm enhanced global search and convergence. Experimental results showed that IWOA outperformed WOA, BA, and ICA on both synthetic and real-world manufacturing data.

A hybrid task scheduling algorithm called CWOA, combining Cuckoo Search Algorithm (CSA) and WOA, was proposed in [11] to address the NP-hard problem of scheduling in cloud environments. The algorithm aims to improve key performance metrics such as makespan, memory utilization, and energy consumption. Compared to CSA, WOA, and Ant Colony Optimization, the CWOA achieved notable improvements across all metrics, demonstrating its effectiveness in enhancing cloud system performance.

2.1 | Research Gap and Novel Contributions

Despite the valuable existing studies, most current hybrid methods employ a fixed structure for algorithm integration and pay limited attention to the dynamic nature of the optimization environment. Moreover, many of these approaches have been tested only on benchmark functions with insufficient evaluation on real-world problems. The present study aims to address this research gap by proposing a flexible hybrid framework that combines PSO and WOA. The main innovation lies in the use of an intelligent switching mechanism based on population diversity and convergence criteria, which dynamically balances the exploration and exploitation phases.

3 | Problem Statement

Task scheduling in cloud computing environments is considered one of the fundamental challenges in resource management. In this study, we address the problem of assigning a set of N independent tasks to a set of M heterogeneous virtual machines. The primary objective is to minimize the overall task completion time (makespan), which is defined as the maximum completion time among all virtual machines. In fact, in this problem, we denote the overall completion time of tasks as L , and the goal is to minimize L . Eq. (1) presents the objective function for our problem.

$$\min L. \quad (1)$$

As mentioned above, the machines in our problem are defined as heterogeneous. The heterogeneity of machines here means that their processing capabilities differ from one another. Therefore, we define C_j as the capacity of machine j (for $1 \leq j \leq M$). On the other hand, the tasks to be processed also have different processing times. Hence, t_i is considered as the processing time of the i -th task (for $1 \leq i \leq N$).

To assign a task to a machine, we define the binary variable x_{ij} as follows: if $x_{ij} = 1$, then the i -th task is assigned to the j -th machine; otherwise, its value is 0. Based on the above explanation, the problem can be modeled using the following linear programming formulation:

$$\begin{aligned}
& \min \quad L, \\
& \text{s.t.} \quad \sum_{j=1}^M x_{ij} = 1, \text{ for all } i = 1 \dots N, \\
& \quad \sum_{i=1}^N \frac{t_i}{c_j} x_{ij} \leq L, \text{ for all } j = 1 \dots M, \\
& \quad x_{ij} = \{0,1\}, \text{ for all } i = 1 \dots N, \text{ for all } j = 1 \dots M.
\end{aligned} \tag{2}$$

Since the processing capabilities of the machines differ, the processing time of the i^{th} task on the j^{th} machine is considered as $\frac{t_i}{c_j}$. The first constraint ensures that each task is assigned to exactly one machine. The second constraint states that the total execution time on each machine must not exceed L . The third constraint defines the variable x_{ij} as binary, which means that it can only take values of 0 or 1.

In the considered model, it is assumed that the tasks are completely independent, with no dependencies or the need for data exchange between them. This characteristic eliminates the complexity arising from communication delays between tasks. The execution time of each task on a specific virtual machine is calculated as the ratio of the task length (in MI) to the processing capacity of the virtual machine (in MI Per Second, MIPS).

This problem belongs to the class of NP-hard problems, as the solution space grows exponentially with the number of tasks and virtual machines. For example, in a system with 100 tasks and 10 virtual machines, the number of possible allocation combinations reaches 10^{100} , making it impossible to find an exact solution in a reasonable amount of time.

The complexity of the problem further increases when considering that virtual machines may become unavailable over time or their processing capacities may change. Moreover, in more realistic scenarios, additional criteria such as energy consumption, execution cost, or Quality of Service (QoS) may also need to be considered, turning the problem into a multi-objective optimization problem.

In cloud computing environments, the optimal allocation of resources has become one of the key challenges. According to recent studies, more than 60% of computing resources in data centers are used inefficiently, leading to increased operational costs and degraded QoS [12].

Recent studies indicate that two main factors contribute to this challenge: on the one hand, the diversity and heterogeneity in the computational requirements of tasks, and on the other hand, the differences in the capacity and capabilities of available resources. This heterogeneity has led to the ineffectiveness of many traditional resource allocation methods in numerous scenarios [13].

In this context, intelligent optimization algorithms have emerged as a promising solution. However, each of these algorithms has its own limitations. For example, although PSO performs well in exploring promising regions of the solution space, it struggles to fine-tune the final solutions [1]. However, while the WOA demonstrates strong local search capabilities, its performance tends to degrade in high-dimensional environments [2].

In this study, a combination of the WOA and PSO is employed to address this problem. The aim of this hybrid approach is to leverage the strengths of both algorithms in order to achieve high accuracy, fast convergence, and to avoid getting trapped in local optima.

3.1| Whale Optimization Algorithm

The WOA is a metaheuristic inspired by the intelligent hunting behavior of humpback whales. This algorithm simulates the “bubble-net” feeding strategy and models three main behaviors:

- I. Encircling the prey: mimicking exploitation of the current best solution.
- II. Spiral attacking: a spiral-shaped movement toward the prey.
- III. Searching for new prey: to enhance population diversity and avoid local optima.

WOA provides a good balance between global exploration and local exploitation, making it a powerful approach for solving complex optimization problems. However, in some cases, it may converge more slowly compared to other population-based algorithms.

In this problem, we are given N independent tasks to be assigned to M virtual machines. Therefore, each whale (i.e., solution) can be represented by an array of length N :

$$X = [x_1, x_2, x_3, \dots, x_n]. \quad (3)$$

where $x_i \in \{1, 2, \dots, M\}$ denotes the machine to which task i is assigned. For instance, if we have $M = 3$ machines and $N = 5$ tasks, a solution vector $X = [2, 3, 1, 1, 2]$ indicates that:

- I. Task 1 is assigned to machine 2.
- II. Task 2 is assigned to machine 3.
- III. Tasks 3 and 4 are assigned to machine 1.
- IV. Task 5 is assigned to machine 2.

The goal is to minimize the overall completion time (Makespan), which is defined as the maximum execution time among all machines. The total processing time on machine j is computed as

$$T_j = \sum_{i=1}^N \frac{t_i}{c_j} \cdot \delta_{ij}, \quad (4)$$

where:

- I. t_i is the length (in MI) of task i .
- II. c_j is the processing capacity (in MIPS) of machine j .
- III. $\delta_{ij} = \begin{cases} 1 & \text{if } x_i = j, \\ 0 & \text{otherwise.} \end{cases}$
- IV. Then, the fitness function $f(x)$ is defined as

$$f(x) = \max_{1 \leq j \leq M} T_j, \quad (5)$$

The goal of the optimization process is to minimize $f(x)$. The pseudocode of the WOA for solving the task scheduling problem on machines can be seen in *Algorithm 1*.

Algorithm 1. Whale Optimization Algorithm (WOA) for Task Scheduling.

```

Input: Task sizes  $T = \{t_1, \dots, t_n\}$ , machine speeds  $M = \{m_1, \dots, m_k\}$ , number of iterations
 $I$ , number of whales  $W$ 
Initialize  $W$  whales with random assignments of  $n$  tasks to  $k$  machines
Evaluate fitness (makespan) of each whale
Set the best whale as  $X^*$ 
for each iteration  $i = 1$  to  $I$  do
   $a \leftarrow 2 - 2 \cdot \frac{i}{I}$ 
  for each whale  $X$  in population do
    for each task  $j$  do
      Generate random numbers  $r_1, r_2 \in [0,1]$ 
      Compute  $A = 2ar_1 - a, C = 2r_2$ 
      if  $|A| < 1$  then
        Update  $X_j \leftarrow \text{round}(X_j^* - A \cdot |CX_j^* - X_j|)$ 
      else
        Select random machine rand
        Update  $X_j \leftarrow \text{round}(\text{rand} - A \cdot |\text{rand} - X_j|)$ 
      end if
    end if
    Clamp  $X_j$  to  $[0, k - 1]$ 
  end for
  Evaluate new fitness
  if new fitness better than  $X^*$  then
    Update  $X^* \leftarrow X$ 
  end if
end for
end for
Output: Best task-to-machine assignment  $X^*$  and its makespan

```

3.2| Particle Swarm Optimization Algorithm

PSO is a metaheuristic algorithm inspired by the social behavior of birds and fish, where each particle represents a potential solution to the problem. Particles move through the search space, and their velocity and position are updated based on their own individual experience as well as the collective experience of the swarm. More specifically, the velocity and position of each particle are updated in each iteration by combining the particle's personal best position and the global best position found by the entire swarm. PSO exhibits fast convergence speed; however, it can be prone to getting trapped in local optima, especially when the diversity of the swarm decreases.

To solve this problem using PSO, each particle's position represents a solution, where each particle has a position and a velocity. In this problem, the positions must be mapped to discrete values, i.e., the machine indices. Finally, the personal best position (pBest) and the global best position (gBest) are used to guide the search. The pseudocode of the PSO algorithm is shown in *Algorithm 2*.

3.3| Hybrid WOA-PSO Algorithm

The combination of WOA and PSO has been carried out to leverage the advantages of both algorithms. PSO has a high capability in quickly finding suitable solutions, while WOA possesses a strong ability to avoid getting trapped in local optima through its random search mechanisms and spiral attack behavior. By combining these two, an algorithm can be achieved that has both a high convergence speed and the ability to escape from non-optimal regions. In the hybrid approach, the algorithm is divided into two phases. The first phase involves strong exploration, which is performed by the PSO algorithm. In fact, PSO quickly discovers promising regions of the search space. This phase enhances diversity and helps to avoid getting trapped in local optima. The second phase uses the WOA algorithm for precise exploitation. The best solutions found

by PSO are used as the initial population for WOA. Then, WOA operates more accurately to optimize the final solution. This process leads to more precise local exploitation. Since the data structure of the solutions is the same and the fitness function is similar in both algorithms, it is sufficient to run the first part of the iterations using PSO and complete the final iterations using WOA. The pseudocode for this approach is shown in *Algorithm 3*.

Algorithm 2. PSO for task scheduling.

Input: Task sizes $T = \{t_1, \dots, t_n\}$, machine speeds $M = \{m_1, \dots, m_k\}$, number of particles P ,
max iterations I
Initialize P particles with random task-to-machine assignments
Initialize velocities randomly for each particle
Evaluate fitness of all particles
Set personal best (pbest) for each particle
Set global best (gbest) among all pbest
for each iteration $i = 1$ to I do
 for each particle p do
 Update velocity:

$$v_j \leftarrow w \cdot v_j + c_1 r_1 (pbest_j - x_j) + c_2 r_2 (gbest_j - x_j)$$

 Update position:

$$x_j \leftarrow \text{round}(x_j + v_j) \bmod k$$

 Evaluate new fitness
 if new fitness better than pbest then
 Update pbest
 end if
 if new fitness better than gbest then
 Update gbest
 end if
 end for
 Print best makespan of this iteration
end for
Output: Best task-to-machine assignment (gbest) and its makespan

Algorithm 3. Hybrid PSO-WOA Task Scheduling.

Input: Task times T , machine speeds M , particles P , iterations I_{PSO}, I_{WOA}
Initialize P particles (random assignments) and velocities
Evaluate fitness of each particle
Store personal bests (pbest) and global best (gbest)
for $i = 1$ to I_{PSO} do
 for each particle do
 Update velocity and position
 Evaluate fitness and update pbest, gbest if improved
 end for
end for
Select top K particles from pbest as whales
for $i = 1$ to I_{WOA} do
 Compute $a = 2 - 2i/I_{WOA}$
 for each whale do
 Update position based on best whale or random search
 Evaluate fitness and update best whale
 end for
end for
Output: Best assignment and makespan

4 | Experiments and Results

To evaluate the performance of the proposed hybrid WOA-PSO algorithm in the cloud computing task scheduling problem, the CloudSim simulator was utilized. CloudSim is one of the most widely used simulation frameworks in the cloud computing domain, enabling accurate modeling of data centers, virtual machines, tasks, and scheduling policies.

4.1 | Simulation Configuration

In the simulation environment, tasks and virtual machines are defined with specific characteristics. Task lengths are generated randomly and uniformly within the range of 5000 to 20000 Million Instructions (MI). The processing power of virtual machines varies between 30 and 200 MI Per Second (MIPS). The number of tasks in different scenarios ranges from 100 to 1000, while the number of virtual machines varies between 9 and 23. All tasks are assumed to be independent, without any data or temporal dependencies.

4.2 | Algorithm Configuration

To ensure optimal performance of the hybrid WOA-PSO algorithm, the parameter settings were determined through preliminary trial-and-error experiments. Key parameters such as population size, learning rates for PSO, spiral coefficient for WOA, and the maximum number of iterations were carefully adjusted. These settings were chosen with the goal of balancing convergence speed and solution accuracy, while also minimizing execution time. This fine-tuning process plays a crucial role in adapting the algorithm to different task scheduling scenarios in cloud computing environments.

4.3 | Evaluation

To evaluate and compare the performance of the proposed hybrid WOA-PSO algorithm with other classical algorithms such as PSO, GA, and hybrid PSO-GA, several performance metrics were employed:

- I. Makespan: defined as the maximum completion time among all virtual machines. The main objective to minimize this value.
- II. Load balancing: measured using the standard deviation of task execution times across virtual machines, indicating how evenly the workload is distributed.
- III. convergence speed: refers to the rate at which the algorithm reaches the best solution during iterations, reflecting its efficiency in exploring the solution space.

Each experiment was repeated multiple times and the average results were reported to ensure the stability and robustness of the algorithm.

As illustrated in *Fig. 1*, the total execution time across different task sets is shown. Initially, 100 tasks were executed on 9, 12, 18, and 23 virtual machines in *Fig. 1(a)*. Among all algorithms, PSO exhibited the worst execution time. In contrast, the hybrid PSO-WOA algorithm demonstrated the best performance. This indicates that although PSO alone performs poorly, it plays a significant role in initializing WOA with better parameters. When the output solutions from PSO are passed as the initial population to the WOA algorithm, the performance of WOA improves, leading to better optimization results.

Subsequently, the number of tasks was increased and experiments were also conducted for 300, 600, and 1000 tasks (shown in *Figs. 1(b)* and *1(d)*). The results demonstrate that the hybrid PSO-WOA algorithm consistently achieves the best performance among all compared algorithms. The PSO-GA hybrid algorithm, on the other hand, shows mixed results: in some cases, it provides good solutions, while in others, it delivers the worst performance. Therefore, the PSO-GA combination cannot be considered a reliable optimization method for this task scheduling problem.

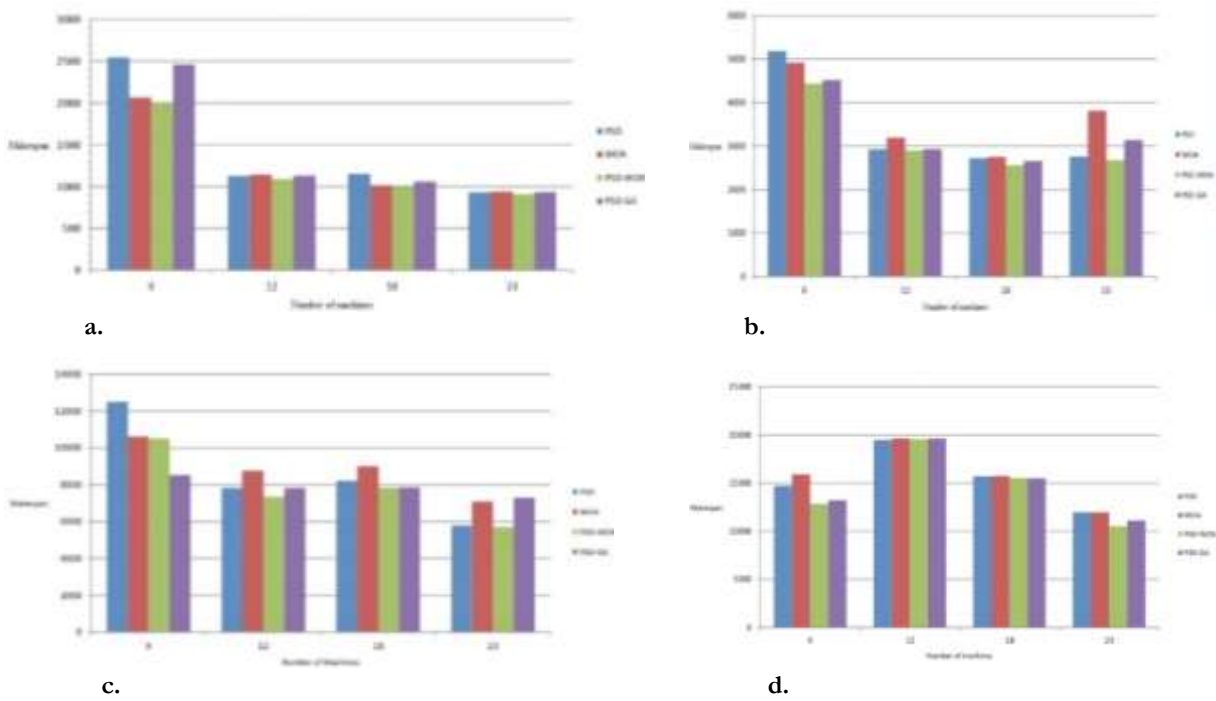


Fig. 1. Makespan across different task numbers; a. Makespan for 100 tasks, b. Makespan for 300 tasks, c. Makespan for 600 tasks, d. Makespan for 1000 tasks.

Fig. 2 illustrates the performance of the PSO-WOA hybrid algorithm executed on 23 virtual machines with a total of 1000 tasks. This experiment focuses on the impact of the initial population size. The algorithm was tested with different initial population sizes, and the total execution time was recorded. For each specific population size, the algorithm was run five times, and the average of the total execution times was taken as the final result.

As shown in Fig. 2, when the initial population size is small, the algorithm performs poorly and results in a higher total execution time. As the population size increases, the execution time improves. According to the results, an initial population size of 50 yields the best performance in terms of total execution time. Beyond this point, increasing the population size does not improve the performance; in fact, the total execution time begins to degrade. After the population size exceeds 90, the algorithm reaches a relatively stable state.

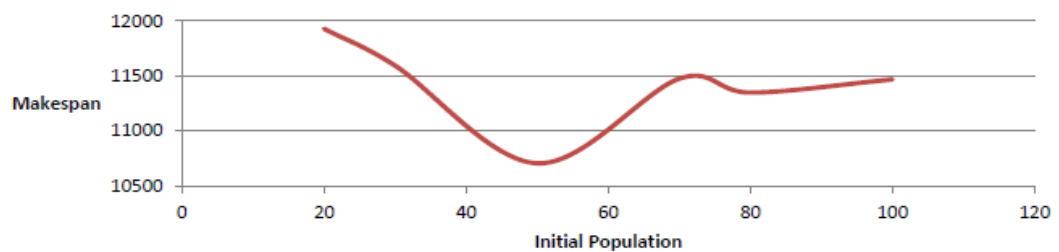
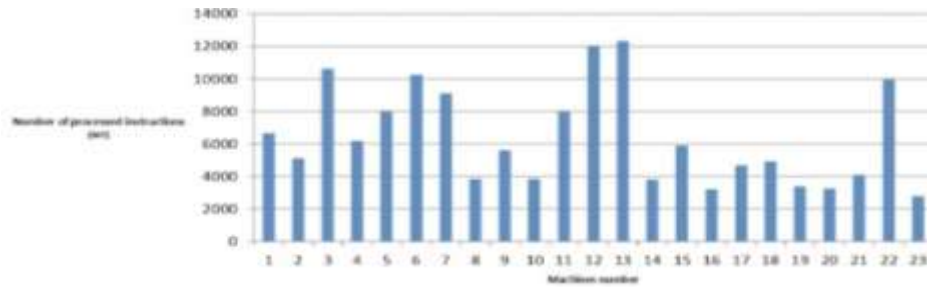


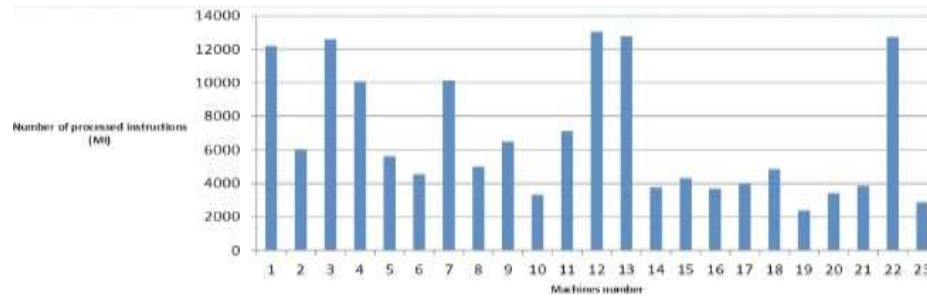
Fig. 2. Makespan with different initial populations.

However, the optimal performance is clearly observed at a population size of 50.

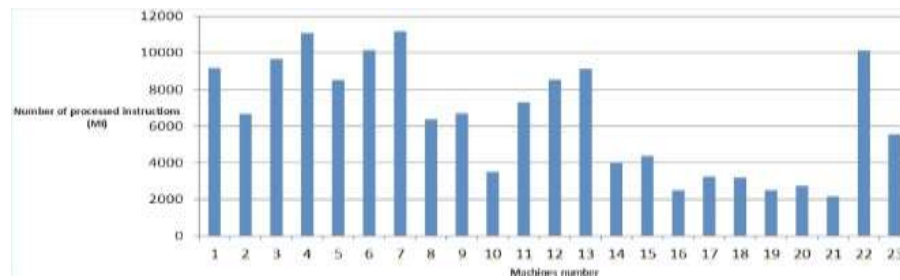
Fig. 3(a) to 3(d) illustrate the distribution of 1000 tasks across 23 virtual machines for each of the evaluated algorithms. Based on these distributions, the standard deviation in the hybrid PSO-WOA algorithm is 2949, a more balanced load, suggesting that the algorithm effectively utilizes all available machines more uniformly.



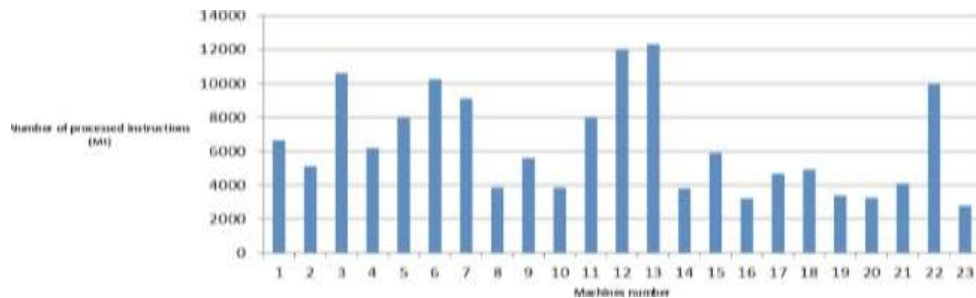
a.



b.



c.



d.

Fig. 3. Task allocation among machines; a. WOA-PSO algorithm, b. PSO-GA algorithm, c. PSO algorithm, d. WOA algorithm.

As observed in Fig. 3.a, the number of instructions processed by each of the 23 machines is relatively balanced, except for a few machines that handle slightly more workload. This reflects a more uniform task distribution across computing resources, which leads to increased overall resource efficiency and improved algorithm utilization.

Table 1. Convergence of algorithms on 1000 tasks and 23 machines.

Algorithms	Convergence
PSO	31
WOA	39
PSO-WOA	80
PSO-GA	93

In the final evaluation, shown in *Table 1*, the convergence behavior of each algorithm is analyzed. In this experiment, each algorithm was executed on a scenario with 1000 tasks and 23 virtual machines, with the total number of iterations set to 100. As presented in *Table 1*, the PSO algorithm demonstrates the fastest convergence compared to the others. It is followed by WOA, which also performs well in reaching near-optimal solutions. The hybrid algorithms, PSO-WOA and PSO-GA, rank next in terms of convergence speed.

In this evaluation, the PSO-WOA hybrid algorithm did not outperform the standalone PSO and WOA algorithms in terms of convergence speed. This outcome can be attributed to two main reasons: 1) Since hybrid algorithms consist of two sequential phases, their convergence cannot be expected to occur in less than half of the total number of iterations. Otherwise, the use of a hybrid structure becomes ineffective. 2) Although the PSO algorithm achieves convergence in fewer iterations and requires less execution time, this may indicate that it quickly becomes trapped in a local optimum and fails to escape. Therefore, while PSO converges rapidly, it does not necessarily guarantee better final solutions.

5 | Conclusion and Future Work

In this study, a novel hybrid approach for task scheduling in cloud computing environments was proposed by combining the WOA with PSO. The proposed WOA-PSO algorithm was designed to enhance key performance metrics such as minimizing the makespan, improving resource utilization, and achieving better load balancing.

Simulation results conducted using the CloudSim environment demonstrated that the WOA-PSO algorithm outperforms baseline algorithms such as PSO, WOA, and the hybrid PSO-GA in various scenarios, consistently achieving shorter task completion times. This superior performance is attributed to the intelligent integration of PSO's fast convergence ability and WOA's strong exploratory capabilities, which help avoid local optima. In addition to reducing execution time, the proposed algorithm effectively maintains load balance across virtual machines, contributing to improved QoS in cloud systems.

For future work, enhanced versions of WOA or PSO can be explored, or more advanced multi-stage hybrid algorithms can be designed. Additionally, incorporating task dependencies and data transfer costs could be investigated to further improve the effectiveness of the scheduling system.

Reference

- [1] Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of icnn'95-international conference on neural networks*. Vol. 4, pp. 1942–1948. <https://doi.org/10.1109/ICNN.1995.488968>
- [2] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, 95, 51–67. <https://doi.org/10.1016/j.advengsoft.2016.01.008>
- [3] Akbari, M., Rashidi, H., & Alizadeh, S. (2017). An enhanced genetic algorithm with new operators for task scheduling in heterogeneous computing systems. *Engineering applications of artificial intelligence*, 61, 35–46. <https://doi.org/10.1016/j.engappai.2017.02.013>
- [4] Ibrahim, M. A., Al-Tahar, I. A., Salamah, H. M., & Mohamad, N. I. (2024). Improving quality of service in cloud computing frameworks using whale optimization algorithm. *ingénierie des systèmes d'information*. <https://api.semanticscholar.org/CorpusID:273607414>

- [5] Kao, Y.-T., & Zahara, E. (2008). A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied soft computing*, 8(2), 849–857. <https://doi.org/10.1016/j.asoc.2007.07.002>
- [6] Chhabra, A., Huang, K.C., Bacanin, N., & Rashid, T. A. (2022). Optimizing bag-of-tasks scheduling on cloud data centers using hybrid swarm-intelligence meta-heuristic. *The journal of supercomputing*, 78(7), 9121–9183. <https://doi.org/10.1007/s11227-021-04199-0>
- [7] Zhang, Y., & Wang, J. (2024). Enhanced whale optimization algorithm for task scheduling in cloud computing environments. *Journal of engineering and applied science*, 71(1), 121. <https://doi.org/10.1186/s44147-024-00445-3>
- [8] Strumberger, I., Bacanin, N., Tuba, M., & Tuba, E. (2019). Resource scheduling in cloud computing based on a hybridized whale optimization algorithm. *Applied sciences*, 9(22), 4893. <https://doi.org/10.3390/app9224893>
- [9] Chen, X., Cheng, L., Liu, C., Liu, Q., Liu, J., Mao, Y., & Murphy, J. (2020). A WOA-based optimization approach for task scheduling in cloud computing systems. *IEEE systems journal*, 14(3), 3117–3128. <https://doi.org/10.1109/JSYST.2019.2960088>
- [10] Han, Z., Zhang, Q., Shi, H., Qi, Y., & Sun, L. (2019). Research on limited buffer scheduling problems in flexible flow shops with setup times. *International journal of modelling, identification and control*, 32(2), 93–104. <https://doi.org/10.1504/IJMIC.2019.102360>
- [11] Pradeep, K., Ali, L. J., Gobalakrishnan, N., Raman, C. J., & Manikandan, N. (2022). Cwoa: Hybrid approach for task scheduling in cloud environment. *The computer journal*, 65(7), 1860–1873. <https://doi.org/10.1093/comjnl/bxab028>
- [12] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A. (2010). A view of cloud computing. *Communications of the acm*, 53(4), 50–58. <https://dl.acm.org/doi/pdf/10.1145/1721654.1721672>
- [13] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future generation computer systems*, 25(6), 599–616. <https://doi.org/10.1016/j.future.2008.12.001>