Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 22 No. 7 (2026) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v22i07.61591

PAPER

An Efficient Task Scheduling Approach in Cloud Computing Using Hybrid Fruit Fly and Ant Colony Optimization Techniques

Narayana Rao Appini[1] (✉), K. Premnadh[2], Karnam Sreenu[3], Prasadu Gurram[2], Siddabathuni Suresh Babu[1]

[1]N.B.K.R. Institute of Science and Technology, Tirupati, Andhra Pradesh, India

[2]Sreenidhi Institute of Science and Technology, Ghatkesar, Telangana, India

[3]Aditya University, Kakinada, Andhra Pradesh, India

narayanaraoappini@nbkrist.org

ABSTRACT

Although cloud computing offers on-demand resources, effective job scheduling is still a major challenge to increase efficiency and resource usage. For large-scale, dynamic workloads, traditional algorithms s FCFS and Min-Min are straightforward yet ineffective. In order to generate high-quality task–VM mappings, this study suggests a hybrid nature-inspired algorithm called fruit fly optimization–ant colony optimization (FOA-ACO), which combines the exploitative ant colony optimization (ACO) and the exploratory fruit fly optimization algorithm (FOA). Minimizing makespan, optimizing resource usage, and distributing load evenly among virtual machines (VM) are the objectives. Workload traces from PlanetLab and CloudSim 3.0.3 are used to assess the method in a variety of experimental scenarios involving up to 1000 jobs and numerous diverse VM. The suggested FOA-ACO methodology enhances overall cloud performance by decreasing the makespan by 9.57%, employing more resources by 7.14%, and enhancing the load balancing factor (LBF) by 23.06%. This was observed by comparing it with other approaches such as FCFS, Min-Min, solo ACO, and WOA. An effective solution to the scheduling problems in today's cloud environments is provided by this hybrid method. Future research may concentrate on refining the system to take into consideration various objectives, such as cost and reliability, enhancing its use of energy, and optimizing its performance with very massive cloud setups. Also, the approach might be refined to facilitate decision-making in real time and incorporate machine learning methods for greater flexibility.

KEYWORDS

cloud computing, task scheduling, fruit fly optimization, ant colony optimization (ACO), whale optimization algorithm (WOA), CloudSim, PlanetLab

Appini, N. R., Premnadh, K., Sreenu, K., Gurram, P., Babu, S. S. (2026). An Efficient Task Scheduling Approach in Cloud Computing Using Hybrid Fruit Fly and Ant Colony Optimization Techniques. *International Journal of Online and Biomedical Engineering (iJOE)*, 22(7), pp. 52–66. https://doi.org/10.3991/ijoe.v22i07.61591

Article submitted 2026-01-17. Revision uploaded 2026-03-15. Final acceptance 2026-03-21.

1 INTRODUCTION

Although cloud computing facilitates on-demand, scalable, and cost-effective online access, it has changed the way we use computer resources. It has evolved to be a major advancement in technology and is vital for many different kinds of disciplines, particularly scientific research, e-commerce, and the organization of massive volumes of data. The necessity of scalable cloud settings was first highlighted by [1]. Later, [2] introduced CloudSim, a well-known program for simulating cloud systems, managing resources, and workload distribution. Effective scheduling tasks become difficult in Infrastructure-as-a-Service (IaaS) clouds due to workloads regularly vary and resources are not uniform [3], [4].

The scheduling of tasks is the procedure of assigning user tasks to unused VMs in order to boost performance measures such as balancing load, time for task completion reduction, efficiency of resources, and conservation of energy [5], [6]. Even though standard scheduling techniques such as Min-Min and First-Come-First-Served (FCFS) are rapid, they are incapable of adapting to the complexity of numerous and various environments [7]. This has led to the widespread adoption of metaheuristic techniques such as genetic algorithms (GA) [8], particle swarm optimization (PSO) [9], ant colony optimization (ACO) [10], and whale optimization algorithm (WOA) [12].

These approaches may result in almost optimal solutions and are efficient at dealing with NP-hard scheduling problems. Hybrid metaheuristics, which incorporate multiple algorithms to balance developing fresh approaches and strengthening ones that already exist, have received more focus in recent studies [13], [15]. As an instance, even though WOA and other swarm-based techniques are efficient at providing global solutions, they difficulty with local fine-tuning. ACO, on the contrary hand, has the capability of refining solutions, however it might end prematurely [14]. The issue lies with developing a mixed approach that effectively addresses cloud scheduling by utilizing the beneficial features of different methods [16]. In addition, it matters to decide on workloads that precisely reflect real-world scenarios for the purpose of confirming scheduling strategies. PlanetLab traces is able to develop realistic and varied task sets, as illustrated by Lin et al. [17], [18]. How to thoroughly test these models in order to guarantee that they are reliable is addressed by [19].

These findings help this study, which uses CloudSim and PlanetLab data to thoroughly test the proposed algorithm. This study introduces a new combined method called fruit fly optimization–ant colony optimization (FOA-ACO), which mixes the ability of the fruit fly optimization algorithm (FOA) to explore (exploration) solutions with the strength of ACO in finding the best (exploitation) options. The main goals are to reduce the total time (makespan) needed to complete tasks, use resources (resource utilization) more efficiently, and ensure that work is spread evenly across VMs. To prove its superiority under actual cloud workloads, the suggested method is empirically evaluated against FCFS, Min-Min, ACO, and whale optimization algorithm.

With the rapid development of cloud computing, effective task scheduling has become a key challenge because of the scale, heterogeneity, and dynamics of cloud resources and workloads, and cloud service providers need to process large numbers of user tasks with strict performance standards, such as balanced workload allocation, high resource utilization, and short execution times. Inefficient scheduling affects service quality and customer satisfaction, leading to resource underutilization, increased makespan, performance degradation, and higher operating costs. Traditional scheduling algorithms are unable to adapt to dynamic and

heterogeneous contexts, leading to VM imbalance and poor scalability. As cloud infrastructures develop and serve various applications such as big data analytics, IoT, and real-time services, the need for intelligent and adaptive task scheduling mechanisms increases.

Organization: Section 2 describes related work. Section 3 discusses system architecture. Section 4 provides a description of the suggested methodology. Section 5 discusses the simulation setup. Section 6 part provides a description of the findings. Conclusion and future efforts are covered in Section 7.

2 LITERATURE SURVEY

[1] Introduced the basic principles for scalable cloud systems, laying the foundation for modern cloud architecture. Building on this, [2] created CloudSim, a simulation toolset that is now widely used for evaluating and contrasting scheduling algorithms. Li and Buyya [3] expanded these frameworks to include dynamic scheduling capabilities, making them suitable for heterogeneous and large-scale cloud systems.

The complexity of dynamic workloads is too much for traditional heuristic algorithms like FCFS and Min-Min, notwithstanding their speed and simplicity [7]. The first metaheuristic framework for NP-hard scheduling issues using GA was presented by [8]. Although PSO-based methods, like the research of [9], provide better convergence, they frequently become stuck in local optima. These limitations necessitate the use of hybrid techniques.

[10] Was inspired by the foraging behaviors of ants to create ACO. [11] Shown improvements in makespan and energy efficiency by using ACO in cloud work allocation. However, because ACO's efficacy is largely dependent on proper pheromone start and parameter tweaking, it is vulnerable to early convergence.

[12] Proposed WOA, which mimics the bubble-net hunting behavior of humpback whales. [14] Showed that WOA can enhance exploration, but it is limited in its exploitation performance in fine-tuning task assignments. There has been an increased focus on hybrid approaches that combine the strengths of multiple metaheuristics [13], [15]. A hybrid approach of GA-ACO by [21] achieved a better solution variety and convergence speed. [13] reviewed hybrid scheduling algorithms and the need for the exploration-exploitation trade-off, which has also been covered in optimization techniques by [16].

[24–26] Used three Meta-Heuristic algorithms for efficient work scheduling: WOA, fractional GWO (FGWO), and modified fractional GWO (MFGWO). In task scheduling, FGWO and MFGWO used several metrics, such as execution cost, resource usage, communication time, energy, and communication cost, while WOA formed the fitness function based on the budget cost and makespan parameters. [17] and [18] highlighted the importance of realistic workloads (e.g., PlanetLab traces) to evaluate cloud scheduling methods, and [19] provided guidelines for cloud performance benchmarking to ensure the integrity of the experiments. [20] identified significant patterns and open research problems in their metaheuristic scheduling analysis.

[27] Proposed a simple task scheduling mechanism based on multiple resources attributes and conditional logic, which frame the conditional logic using the resource parameters such as reliability, availability, reaction time, and location. The scheduling index is measured based on resource parameters to determine the best

resources for task scheduling with three objectives of faster convergence, minimum makespan, and maximum throughput.

[28] Proposed an enhanced social learning optimization–based task scheduling method for cloud computing to reduce makespan and improve resource utilization. The approach enhances overall efficiency of task execution in cloud environments. [29] Introduced a hybrid resource allocation model combining Grey Wolf Optimizer and Artificial Bee Colony algorithms. The method improves load balancing and optimizes cloud resource utilization, leading to better system performance.

Recent research by [6] and [22] places a strong emphasis on maximizing energy efficiency and resource use. However, the majority of algorithms do not specifically handle workload balance, instead giving priority to makespan or cost. Dynamic scheduling techniques explored by [4] and [23] indicate a growing need for adaptive, scalable solutions. Despite progress, hybrid methods with dynamic workload adaptation remain underexplored.

The gaps found in the literature are, Limited multi-objective optimization, in which the majority of research concentrates on a single measure while ignoring the concurrent optimization of load balancing, makespan, and resource utilization. Due to a poor balance between exploration and exploitation, standalone FOA or ACO algorithms have flaws that frequently result in premature convergence and less-than-ideal outcomes. Due to their lack of a systematic design, hybrid techniques are unable to successfully integrate local refining with global search. ACO-based models lack adaptive balancing factors and dynamic pheromone initialization, which are essential for quicker convergence and a wider range of solutions. Limited capacity to adjust to changing cloud settings, which reduces the effectiveness of current algorithms under workloads that are varied and unpredictable. reliance on artificial workloads as opposed to real-world datasets, such as PlanetLab, which causes a disconnect between experimental assessment and practical application.

The FOA-ACO hybrid algorithm, which has the following contributions, is proposed in this paper to fill in these gaps:

- A new pheromone initiation technique that makes use of FOA's capacity for wide exploration.
- ACO integration for solution refining and fine-grained exploitation.
- Three important metrics-makespan, resource usage, and load balance are explicitly optimized.
- Comprehensive evaluation using CloudSim and PlanetLab workloads to validate performance against state-of-the-art algorithms.
- Realistic evaluation using 1,000 heterogeneous tasks and 50 VMs configured in CloudSim.
- Empirical evidence showing FOA-ACO outperforms FCFS, Min-Min, pure ACO, and whale optimization algorithm.

Recent research in cloud task scheduling has increasingly focused on metaheuristic and hybrid optimization techniques to address the limitations of traditional heuristics such as FCFS and Min-Min. While algorithms like ACO, WOA, and GA improve scheduling performance, many studies optimize limited objectives, suffer from convergence issues, or lack validation using realistic workloads. Inspired by these shortcomings, this work tackles the challenge of creating an effective hybrid scheduling system that maximizes resource usage, decreases makespan, and enhances load balancing in heterogeneous cloud environments.

This work's main contribution is a hybrid FOA-ACO task scheduling framework that simultaneously optimizes load balancing, makespan, and resource utilization.

According to CloudSim simulations, the suggested method outperforms FCFS, Min-Min, ACO, and WOA in terms of convergence and scheduling efficiency by fusing the global exploration of FOA with the exploitation capacity of ACO. The suggested FOA-ACO technique has some drawbacks despite its efficacy. All real-world cloud dynamics are not captured by the evaluation, which is restricted to CloudSim simulations. Only makespan, resource utilization, and load balancing are taken into account in the study; other QoS measures like energy consumption, cost, and SLA breaches are not.

3 SYSTEM ARCHITECTURE

The suggested FOA-ACO cloud scheduling architecture is shown in Figure 1. It is divided into five different layers, each of which has a specific function to guarantee effective scheduling, resource distribution, and overall cloud performance: User layer (task submission), broker layer (task aggregation and dispatch), scheduler layer (FOA-ACO hybrid engine), VM layer (virtual machines hosting cloudlets), and datacenter layer (physical hosts).

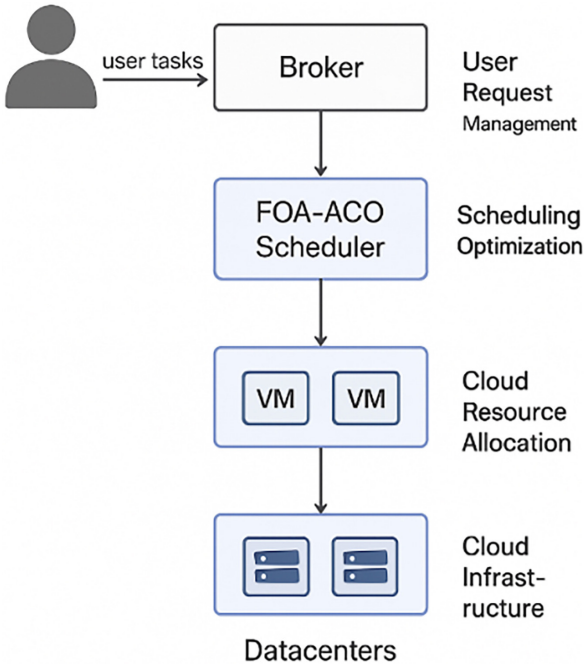

Fig. 1. System architecture of the proposed FOA-ACO task scheduling framework in a cloud computing environment

User layer is the topmost layer where cloud users interact with the system. Users submit tasks or workloads (cloudlets) through a front-end interface. Collects user requests containing information such as task size, computational requirements, and deadlines. Before going into the scheduling pipeline, it guarantees that all workload specifications are accurately recorded and streamlines the task submission process.

Between cloud users and the cloud infrastructure, the Broker Layer serves as a mediator. For processing efficiency, it combines several tasks from different users into batches. Oversees, verifies, and makes sure user requests are prepared for scheduling sends the scheduling engine the prepared task set.

To find possible task-to-VM mappings and steer clear of local optima, the FOA Component (Exploration) in the FOA-ACO Hybrid Engine layer conducts global

exploration. By using the pheromone-driven search to converge on an ideal timetable, the ACO Component (Exploitation) fine-tunes the solution. Lastly, it improves load balancing factor (LBF) and resource consumption while minimizing makespan.

The virtualized computing environment, where cloud resources are abstracted into VMs, is represented by the VM Layer. Predefined resources, such as CPU cores, storage, memory, and bandwidth, are set up in VMs. It guarantees the effective distribution and operation of cloudlets and serves as a link between logical scheduling choices and actual hardware execution.

The real physical infrastructure, such as servers, storage devices, and networking hardware, makes up the datacenter layer. It holds the VM and provides the required computing power manages low-level resource management, storage, and data transport allows for scalability with the addition or removal of physical hosts in response to demand.

4 PROPOSED METHODOLOGY – HYBRID FOA-ACO

The workflow of the suggested FOA-ACO hybrid scheduling methodology for cloud task scheduling is shown in Figure 2.

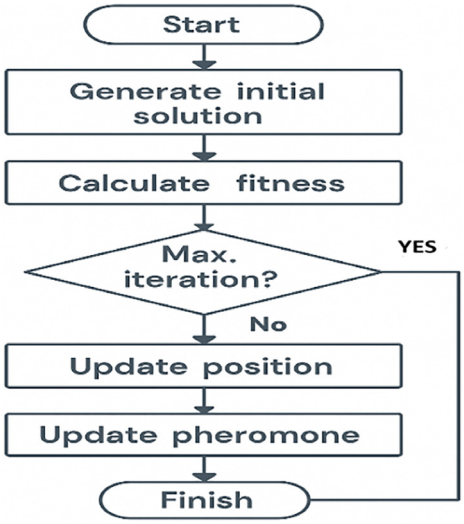

Fig. 2. Workflow of the proposed Hybrid FOA-ACO algorithm

The procedure starts with the creation of the initial solution, in which the FOA is used to create task-to-VM mappings in order to explore the search space broadly. After calculating the performance of each result using important factors like resource utilization, makespan, and load balancing factor, the system checks if the maximum number of iterations has been reached. If not, the method updates the positions by using exploration strategies in FOA to better assign jobs. After that, high-quality solutions are improved using the pheromone update step in ACO, which helps build better scheduling patterns. This process mixes the ability of FOA to explore new options with the ability of ACO to refine and improve solutions, leading to a good final result. When the stopping condition is met, the algorithm stops and gives the best possible task schedule. Compared to more conventional methods like FCFS, Min-Min, standalone ACO, and WOA, this hybrid approach achieves better load balancing, reduces makespan, and improves resource consumption.

Algorithm 1: Hybrid FOA-ACO Task Scheduling

Input: Task set T, VM set VM, FOA parameters (α), ACO parameters (τ, ρ), balancing λ, iterations i.
Output: Mapping of tasks to VMs with minimized makespan

1. Initialize FOA population (N flies)
   generate random solutions
2. Evaluate fitness (smell concentration) of
   each fly using makespan-based objective
3. Initialize ACO pheromone matrix $\tau_{\{ij\}}$ (0)
   using FOA initialization:
   $\tau_{\{ij\}}(0) = \lambda*(1/d_{\{ij\}}) + (1 - \lambda) *FOA_{init}(ij)$
4. for iter = 1 to i do
5. FOA exploration step: update positions,
   evaluate fitness, record FOA_{best}
6. ACO exploitation step: for each ant k
   construct task→VM path using $P_{\{ij\}}$
7. $P_{\{ij\}}(t) = [\tau_{\{ij\}}(t)] \wedge \alpha * [\eta_{\{ij\}}] \wedge \beta/\Sigma_{\{allowed\}}$
8. Update pheromone:
   $\tau_{\{ij\}}(t + 1) = (1 - \rho) *\tau_{\{ij\}}(t) + \Delta\tau_{\{ij\}}$
9. Reinforce pheromones using FOA_{best}:
   $\tau_{\{ij\}} += \gamma * FOA_{best\_contrib}$
10. end for
11. return best mapping found

FOA position update: The following equation updates the position of each fly in FOA. The new position depends on the current best position (X_{best}) and the difference between two random flies. α is the step size control parameter, and rand() function is a random number between [0,1].

$$X_i(t + 1) = X_{best}(t) + \alpha * rand\ () * (X_j(t) - X_k(t)) \quad (1)$$

where $X_i(t)$ is position of fly i at iteration t, $X_{best}(t)$ is best solution found at iteration t.

ACO transition probability: This defines the probability of an ant selecting a particular task-to-VM mapping based on pheromone level (τ) and heuristic information (η). α and β parameters controls the effect of pheromone and experimental values.

$$P_{ij}(t) = ([\tau_{ij}(t)] \wedge \alpha * [\eta_{ij}] \wedge \beta) / \Sigma_{\{allowed\}} ([\tau_{ij}(t)] \wedge \alpha * [\eta_{ij}] \wedge \beta) \quad (2)$$

where $P_{ij}(t)$ is probability of selecting VM j for ith task at 't' iteration, '$\tau_{ij}(t)$' is pheromone intensity for task i → VM j path, η_{ij} is heuristic desirability, often $1/ET_{ij}$, β is influence factor of heuristic information.

Hybrid pheromone initialization: This equation initializes the pheromone level by combining distance-based information and FOA initialization values, balanced by λ

$$\tau_{ij}(0) = \lambda * (1/d_{ij}) + (1 - \lambda) * FOA_{init}(ij) \quad (3)$$

where λ is balancing factor between distance and FOA initialization ($0 \le \lambda \le 1$), d_{ij} is distance or cost metric between ith task and jth VM (e.g., processing time), $FOA_{init}(ij)$: Initial solution quality contribution from FOA.

Execution time (ET) of task i on VM j: This calculates the execution time required for task i to complete on VM j.

$$ET_{ij} = MI_i/MIPS_j \quad (4)$$

where ET_{ij} is execution time of ith task on jth VM (seconds), MI_i is million instructions required by task i (Million Instructions), $MIPS_j$ is million Instructions per second capacity of VM j (Million Instructions/Second).

Makespan: It specifies the maximum completion or execution time among all the VM, indicating the overall length of the schedule.

$$\text{Makespan} = \max(\text{CompletionTime}(VM_j)) \text{ for } j = 1 \text{ to } m \quad (5)$$

where m is the total number of VMs

Load balancing: LBF measures how evenly the workload is distributed across all VMs. A lower LBF indicates better balance.

$$LBF = (\Sigma|Load_j - Load_{avg}|)/(m*Load_{avg}) \quad (6)$$

where $Load_j$ is total processing load on VM j (Million Instructions), $Load_{avg}$ is average load across all VMs (Million Instructions).

5 SIMULATION SETUP

The entire experimental setup leveraging CloudSim 3.0.3 and workload traces from PlanetLab to test the proposed hybrid scheduling algorithm, FOA-ACO (Fruit Fly Optimization with Ant Colony Optimization), is presented in this part. It contains algorithm settings, workload specifics, and environment configuration.

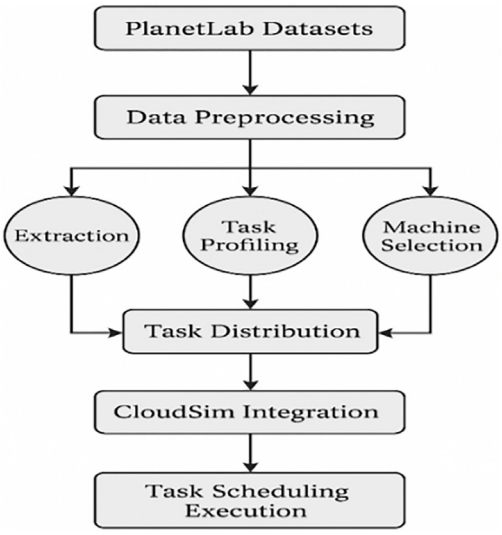

Fig. 3. PlanetLab pre-processing and integration workflow

The experimental procedure for assessing the suggested FOA-ACO scheduling method with CloudSim and PlanetLab datasets is displayed in Figure 3. Real-world workload traces are first collected and pre-processed to clean and structure the data. Next, three key steps take place: extraction of task details, task profiling to analyse task characteristics, and machine selection to profile VMs based on their resources.

The task distribution step comes next, during which ACO fine-tunes various scheduling choices for the best job-to-VM allocation after FOA investigates them. The approach suggested is subsequently implemented in CloudSim, where scheduling is done while performance metrics such as the balance of load factor, consumption of resources, and makespan are determined. Eventually, the outcomes are assessed in relation to other typical methods such as FCFS, Min-Min, ACO, and WOA. This exhibits how FOA-ACO may shorten makespan, enhance resource utilization, and assist in distributing work more fairly in expanding cloud systems. Tables 1 and 2 list the datacenter configuration and algorithm settings, respectively.

5.1 Datacenter/Host/VM configuration

Table 1. Datacenter configuration

Component	Value
Number of Hosts	10
Number of VMs	50
VM MIPS	1000
VM RAM	4096 MB
VM Bandwidth	1000 Mbps
Host RAM	16384 MB
Host Bandwidth	10000 Mbps
Cloudlet Input/Output Size	300 MB/300 MB
Number of Tasks (Cloudlets)	1000
Random Seed	42
Runs per Algorithm	5

5.2 Algorithm parameters

Table 2. Algorithm parameters

Parameter	Value
FOA Population	30
FOA Iterations	6
ACO Ants	20
ACO Iterations	30
λ (pheromone initialization weight)	0.6
γ (FOA reinforcement factor)	0.5

6 RESULTS AND DISCUSSION

In order to find gaps in the current cloud task scheduling techniques, this study started with a thorough literature analysis as part of a methodical research methodology. A hybrid FOA-ACO method was created to simultaneously improve makespan, resource consumption, and load balancing based on these gaps. Objective functions

were established analytically, and a cloud system model was developed. With the help of a hybrid initialization and reinforcement method, the suggested algorithm combines FOA for global exploration with ACO for pheromone-based exploitation. Inspired by PlanetLab traces, heterogeneous workloads were used to implement the technique in the CloudSim simulator.

Experimental simulations were conducted under identical conditions and compared against FCFS, Min-Min, standalone ACO, and WOA algorithms. Performance was evaluated using makespan, resource utilization, and LBF, and the results were analyzed to validate the effectiveness of the proposed FOA-ACO approach.

In relation to makespan, utilization of resources, and distribution of load, the hybrid FOA-ACO algorithm outperformed WOA and traditional ACO. Table 3 displays the simulation results for 1,000 jobs and 50 virtual machines.

Table 3. Simulation results

Algorithm	Makespan	RU	LBF
FCFS	713.949	60.927	0.181
Min-Min	693.746	64.657	0.143
ACO	610.497	68.539	0.101
WOA	573.867	70.356	0.097
FOA-ACO	518.922	75.383	0.075

Makespan: The recommended FOA-ACO strategy's makespan corresponds to the FCFS, Min-Min, ACO, and WOA scheduled tasks approaches in Figure 4. FCFS has the least effective makespan, exhibiting its inefficiency, as it operates tasks linearly without seeking to improve the schedule. Min-Min operates somewhat better by emphasizing on smaller jobs first, but the lack of balance of workloads results in a lengthy makespan. By employing advanced optimization strategies that substantially shorten the makespan, ACO and WOA function effectively, with a smaller makespan, the FOA-ACO approach achieves the best. This reveals how effectively it allocates work, retains a balanced workload, and executes tasks more quickly. These findings show how FOA-ACO outperforms traditional methods, making it an excellent option for cloud-based systems with diverse needs and continuous changes.

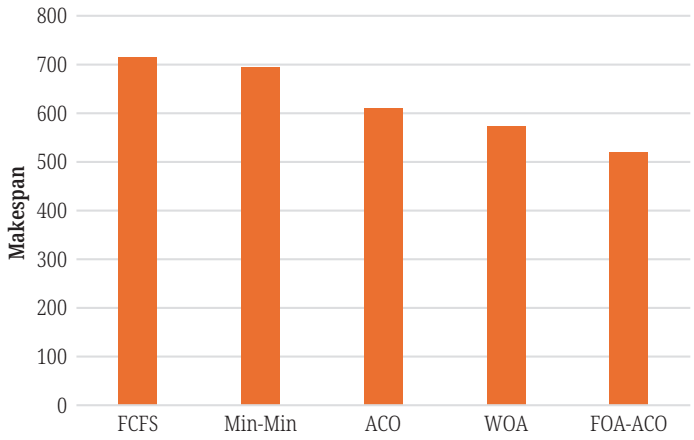

Fig. 4. Makespan comparison of FCFS, Min-Min, WOA, ACO, and FOA-ACO

Resource utilization: In relation of resource utilization, Figure 5 analyzes the recommended FOA-ACO against different scheduling of tasks systems such as FCFS,

Min-Min, ACO, and WOA. Although FCFS sets up jobs in an uncomplicated way, which results in vacant resources, it has the lowest usage of resources. As Min-Min somewhat enhances resource use, improper work allocation remains an issue.

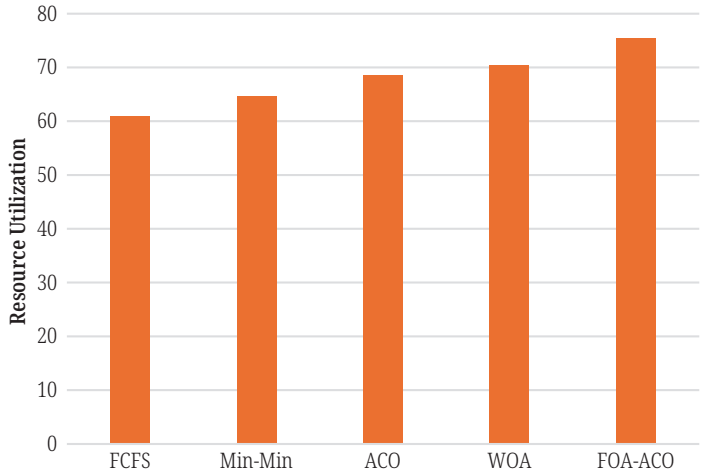

Fig. 5. Resource utilization comparison of FCFS, Min-Min, WOA, ACO, and FOA-ACO

By adopting sophisticated methods for optimization, ACO and WOA function better. However, with regard to of resource use, the FOA-ACO approach operates effectively. It performs this by merging the positive aspects of ACO, which emphasizes the best solutions, with FOA, which explores multiple possibilities. This leads to better distribution of tasks and use of resources, which helps the efficiency of the cloud environment.

Load balancing factor: The computer systems cooperate in the work with greater efficiency when the LBF is smaller. The LBF for multiple methods of scheduling can be found in Figure 6. The outcome suggests that FCFS has the largest LBF, implying that it fails to distribute the work well due to its foundational and inadequate scheduling method. Though Min-Min is a little better, there is yet some imbalance due to the assign's greater workloads to fewer tasks. Because ACO and WOA adopt independent optimization methodologies, they are more effective in distributing the work.

The recommended FOA-ACO strategy yields the shortest LBF through the combination of the analysis of FOA with the implementation of ACO, revealing its efficiency in distributing workloads evenly and optimizing system performance.

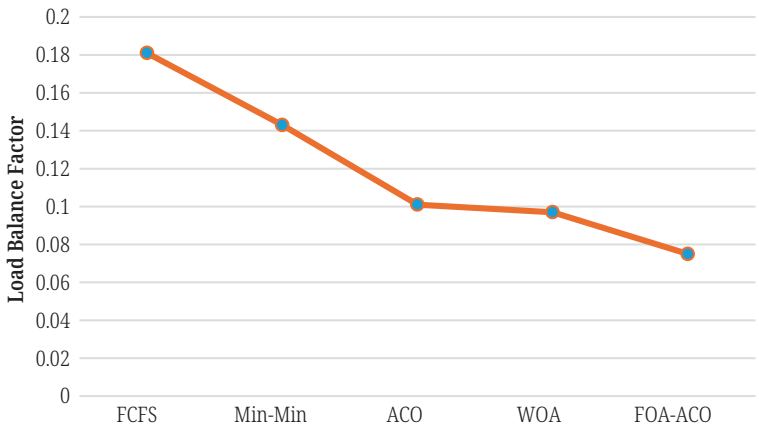

Fig. 6. Load balancing factor comparison of FCFS, Min-Min, WOA, ACO, and FOA-ACO

The following Table 4 shows the comparison of existing methods and proposed methods.

Table 4. Comparative analysis

Aspect	Similarities with Existing Studies	Improvements in FOA-ACO
Scheduling Approach	Uses metaheuristic optimization	Hybrid FOA-ACO instead of standalone
Performance	Reduces makespan as reported in literature	Achieves greater reduction
Resource Utilization	Improves VM utilization	More balanced and higher utilization
Load Balancing	Recognized as important	Explicitly optimized and improved
Convergence Behavior	Metaheuristics improve convergence	Avoids premature convergence
Objective Scope	Limited QoS metrics	Multi-objective optimization
Evaluation Method	Simulation-based	Realistic CloudSim workload modeling

7 CONCLUSION AND FUTURE WORK

This study addressed the challenge of improving cloud task scheduling by jointly minimizing makespan, maximizing resource utilization, and enhancing load balancing. The proposed hybrid FOA-ACO framework effectively combines FOA's global exploration with ACO's exploitation capability, resulting in superior performance over FCFS, Min-Min, ACO, and WOA as demonstrated through CloudSim simulations. The findings confirm that hybrid metaheuristic scheduling achieves better exploration–exploitation balance and improved performance in heterogeneous cloud environments. Future research may concentrate on refining the system to take into consideration various objectives, such as cost and reliability, enhancing its use of energy, and optimizing its performance with very massive cloud setups. Also, the approach might be refined to facilitate decision-making in real time and incorporate machine learning methods for greater flexibility.

8 REFERENCES

[1] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the CloudSim toolkit," in *Proc. IEEE International Conference on High Performance Computing & Simulation (HPCC),* Leipzig, Germany, 2009, pp. 1–11. https://doi.org/10.1109/HPCSIM.2009.5192685

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011. https://doi.org/10.1002/spe.995

[3] X. Li and R. Buyya, "Task scheduling frameworks," *Future Generation Computer Systems*, vol. 137, pp. 123–135, 2022. https://doi.org/10.1016/j.future.2022.04.032

[4] J. Wang, L. Zhou, and M. Huang, "Task scheduling under dynamic workloads," *Journal of Parallel Computing*, vol. 119, pp. 1–15, 2024.

[5] R. Singh and A. Gupta, "Resource allocation in cloud computing," *IEEE Cloud*, vol. 9, no. 2, pp. 145–154, 2023.

[6] Y. Wu and J. Zhang, "Energy-efficient task scheduling in cloud computing," *Future Generation Computer Systems*, vol. 150, pp. 300–310, 2025.
[7] Y. Zhou, X. Wu, and Q. Li, "Heuristic scheduling for multi-cloud environments," *Future Generation of Computer Systems*, vol. 152, pp. 420–433, 2025.
[8] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–72, 1992. https://doi.org/10.1038/scientificamerican0792-66
[9] A. Kumar, D. Verma, and P. Singh, "PSO-based task scheduling in cloud computing," *IEEE Access*, vol. 12, pp. 5689–5698, 2024.
[10] M. Dorigo and T. Stutzle, *Ant Colony Optimization*. Cambridge, MA, USA: MIT Press, 2004. https://doi.org/10.7551/mitpress/1290.001.0001
[11] M. Ahmed, K. Lee, and J. Choi, "Ant colony optimization for task allocation in clouds," *IEEE Transactions on Cloud Computing*, vol. 10, no. 3, pp. 501–510, 2022.
[12] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016. https://doi.org/10.1016/j.advengsoft.2016.01.008
[13] X. Gao *et al.*, "Hybrid metaheuristics in scheduling," *Applied Soft Computing*, vol. 150, p. 109856, 2023.
[14] X. Li, Y. Zhao, and J. Chen, "Whale optimization algorithm for resource scheduling," *Journal of Cloud Computing*, vol. 14, no. 2, pp. 100–113, 2025. https://doi.org/10.1145/3768184.3768228
[15] L. Wang, P. Zhao, and Q. Liu, "Comparative study of metaheuristics," *Applied Soft Computing*, vol. 132, p. 109845, 2024. https://doi.org/10.1016/j.asoc.2022.109845
[16] H. Tang, L. Liu, and Y. Chen, "Exploration vs exploitation in metaheuristics," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC)*, 2024, pp. 220–230.
[17] M. Lin *et al.*, "PlanetLab workload analysis," in *Proceeding of ACM SIGCOMM*, 2023, pp. 151–162.
[18] P. Hu, R. Zhou, and S. Wang, "PlanetLab traces for cloud research," *IEEE Cloud*, vol. 8, no. 1, pp. 67–75, 2024.
[19] S. Patel and M. Dong, "Cloud performance benchmarking," *ACM Computer Surveys*, vol. 57, no. 4, pp. 78–89, 2025.
[20] Y. Wu, X. Zhang, and Z. Li, "Metaheuristic cloud scheduling: A survey," *ACM Computing Surveys*, vol. 57, no. 3, pp. 45–78, 2025.
[21] Z. Zhang, Y. Wang, and X. Liu, "Hybrid GA-ACO for cloud scheduling," *Future Generation Computer Systems*, vol. 147, pp. 231–242, 2023.
[22] R. Singh, S. Patel, and N. Verma, "Improving cloud resource utilization," *IEEE Transactions on Cloud Computing*, vol. 12, no. 4, pp. 700–711, 2024.
[23] A. Khan, M. Ali, and S. Haque, "Dynamic task scheduling approaches," *Future Generation Computer Systems*, vol. 151, pp. 400–412, 2025.
[24] K. Sreenu and M. Sreelatha, "W-Scheduler: Whale optimization for task scheduling in cloud computing," *Cluster Computing*, vol. 22, no. S1, pp. 1087–1098, 2019. https://doi.org/10.1007/s10586-017-1055-5
[25] K. Sreenu and M. Sreelatha, "FGMTS: Fractional Grey Wolf optimizer for multi-objective task scheduling strategy in cloud computing," *Journal of Intelligent and Fuzzy Systems*, vol. 35, no. 1, pp. 831–844, 2018. https://doi.org/10.3233/JIFS-17148
[26] K. Sreenu and M. Sreelatha, "MFGMTS: Epsilon constraint-based modified fractional Grey Wolf optimizer for multi-objective task scheduling in cloud computing," *IETE Journal of Research*, vol. 65, no. 2, pp. 201–215, 2018. https://doi.org/10.1080/03772063.2017.1409087
[27] K. Sreenu and M. Sreelatha, "Multiple resource attributes and conditional logic assisted task scheduling in cloud computing," *International The Intelligent Networks and Systems Society (INASS)*, vol. 16, no. 3, pp. 677–690, 2023. https://doi.org/10.22266/ijies2023.0630.54

[28] Z. Liu, J. Qin, W. Peng, and H. Chao, "Effective task scheduling in cloud computing based on improved social learning optimization algorithm," *International Journal of Online and Biomedical Engineering*, vol. 13, no. 6, pp. 4–21, 2017. https://doi.org/10.3991/ijoe.v13i06.6695

[29] S. Ouhame, Y. Hadi, and A. Arifullah, "A hybrid Grey Wolf optimizer and artificial Bee colony algorithm used for improvement in resource allocation system for cloud technology," *International Journal of Online and Biomedical Engineering*, vol. 16, no. 14, pp. 4–17, 2020. https://doi.org/10.3991/ijoe.v16i14.16623

9 AUTHORS

Dr. Narayana Rao Appini got 18 years of teaching experience and presently is working as a Professor and HOD in the Department of IT and AI&DS, NBKR Institute of Science and Technology, Tirupathi. He cleared UGC NET in 2012 and TS & AP SET 2014. He has published various papers in international journals and conferences. His areas of interests Wireless Networks, Artificial Intelligence, and Data Science. He obtained Ph.D in Computing Science & Engineering, JNTU, Ananthapur, in 2017, India. He received a Master's degree (M.Tech) in Computing Science & Engineering from JNTU, Ananthapur, in 2008, India. He also received his Bachelor of Technology (B.Tech) in Computer Science and Engineering from Narayana Engineering College Gudur, in 2005, India (E-mail: narayanaraoappini@nbkrist.org).

Dr. K. Premnadh is with the Department of Data Science, Sreenidhi Institute of Science and Technology, Yamnampet, Ghatkesar, Telangana, India. He is an Assistant Professor and researcher with over 12 years of academic and technical training experience in reputed engineering institutions. He holds a Ph.D. in Cloud Computing from Annamalai University, along with an M.Tech in Software Engineering and a B.Tech in IT. Currently, he is serving at Sreenidhi Institute of Science and Technology. His research interests include Cloud Computing, Data Science, Distributed Systems, and Cybersecurity. He has actively contributed to research publications, FDPs, and technical training, and is engaged in advancing innovative research in emerging computing technologies (E-mail: premnadh.k@sreenidhi.edu.in).

Dr. Karnam Sreenu is an Assistant Professor in the Department of Information Technology at Aditya University, Andhra Pradesh, with over 17 years of academic experience in reputed engineering institutions. He received his Ph.D. in Computer Science and Engineering from Acharya Nagarjuna University in 2024. He holds an M.Tech in Software Engineering from JNTU Anantapur (2008) and a B.Tech in Information Technology (2005). His research interests include Cloud Computing and Machine Learning, with a focus on developing innovative solutions for emerging computing technologies. He has published several research papers in reputed national and international journals and conferences. He has also actively participated in numerous FDPs, reflecting his commitment to academic excellence, research advancement, and continuous professional development (E-mail: sreenuk@adityauniversity.in).

Dr. Prasadu Gurram working as Assistant Professor, Department of IT, Sreenidhi Institute of Science and Technology (Autonomous), Hyderabad, He is pursuing his Ph.D from Annamalai University, Chidambaram. He completed his M. Tech from Indur Institute of Engineering and Technology, JNTUH, Siddipet. He had Published 5 Papers in National and International Journal and 3 National and International Conference Papers. He had attended 15 workshops, webinars and FDPs. He has 2 Patents and has published two books. He is having 11 years of Experience in

teaching. His areas of Interest are Cloud Computing, Big data, Cyber security and Machine Learning (E-mail: prasadu.g@sreenidhi.edu.in).

Dr. Siddabathuni Suresh Babu has over 26 years of combined teaching and industrial experience. He is currently serving as an Associate Professor in the Department of CSE at NBKR Institute of Science and Technology, India. He has published several research papers in reputed international journals and conferences. His research interests include Wireless Networks, Sensor Networks, and Trust Management Security. He was awarded the Ph.D. in Computing Science and Technology from Sri Krishna Devaraya University, Ananthapur, in 2024. He received his Master of Technology (M.Tech) in CSE from Bharat University, Chennai, in 2009, and his Bachelor of Technology (B.Tech) in ECE from JNTU, Kakinada, in 1999 (E-mail: laksur.suresh@nbkrist.org).