



Multi-Objective Cloud Workflow Scheduling Using Adaptive Multi-Hierarchy Optimization and Ensemble Learning

Deepak Pathak¹, Puneet Dwivedi², Kamal Bunkar³

^{1/2}Department of Computer Science and Engineering, RKDF University, Bhopal, M.P. India,

³Department of Computer Science, Vikram University, Ujjain, M.P. India

ABSTRACT

In this paper an adaptive and hybrid multi-hierarchy optimization (AHMHO) approach is presented for workflow scheduling for cloud computing. The objective is to optimize makespan time, energy consumption, and task distribution across multiple Virtual Machines (VMs). The simulation is conducted using the MATLAB environment. The proposed AHMHO approach is compared against Priority Based and PSO Priority Based task scheduling methods. Results show that the proposed AHMHO method achieves significant improvements in makespan time, lower ADT, stable DI, and reduced SD that makes it more efficient and scalable for cloud-based workflow scheduling for large task sets as well as with increased VMs.

Keywords: Cloud Workflow, Resource Allocation, Virtual Machines, Multi-hierarchy, Optimization.

1. INTRODUCTION

In recent years, cloud computing has transformed how businesses and individuals use computing resources, providing unmatched scalability and flexibility. A key factor in optimizing cloud performance is workflow load scheduling, especially for tasks that are complex and interdependent [1]. Traditional load scheduling methods, which rely on heuristic or rule-based approaches, often struggle to adapt to the changing demands of cloud environments. Machine learning (ML) offers a promising solution by using data-driven insights to make real-time, intelligent decisions. By learning from historical data and adapting to current workloads, ML can enhance resource allocation efficiency, improve scheduling accuracy, and reduce costs [2]-[5]. Cloud computing offers on-demand access to various services for various applications worldwide. It can be configured with public, private, or hybrid cloud deployment models. Most services fall under Infrastructure-as-a-Service, where existing applications can be migrated and deployed in the cloud platform with the support of virtual infrastructure provided by the cloud vendor. [6] These models help provide efficient and effective services to users. Platform-as-a-service allows cloud users to develop applications without installing software, while software-as-a-service allows users to use pre-developed software from various cloud vendors based on their subscription. Both services are available without installation. Scheduling tasks in cloud computing is crucial for workflow applications like data science, image processing, video processing, and big data, which require data association and computational performance. Effective scheduling with appropriate computing resources is crucial for these applications. Workflow scheduling using cloud computing where a decider connects workflow to cloud computing infrastructure, deciding tasks for execution. It sends

decisions to the cloud, where a load balancer assigns tasks to virtual machines (VMs). Cloud providers like AWS, Google Cloud, Azure, and IBM Cloud offer this service. Load balancing is the efficient distribution of network or application traffic across multiple servers to prevent overloading, enhancing application reliability and performance. Traditional methods, based on predefined algorithms, struggle with dynamic systems and demand. Machine Learning (ML) offers intelligent, adaptive, and data-driven approaches to optimize performance and reduce latency. Workflows in cloud environments are structured sequences of automated tasks designed to achieve specific outcomes efficiently and reliably [7]. These workflows comprise individual tasks, their dependencies, required data, and necessary resources. Cloud-native workflows, tailored for scalability and flexibility, leverage cloud services to optimize performance and cost-efficiency. Workflow Management Systems (WMS) like Apache. Airflow and Kubernetes facilitate the orchestration and coordination of these tasks across distributed systems [14]. Cloud workflows support various patterns, including sequential, parallel, and conditional execution, and are essential in applications such as big data analytics, machine learning, and business process automation. Despite their benefits, including scalability and reduced costs, challenges like resource management, latency, and security persist. Emerging trends, such as AI-driven automation and serverless computing, promise to enhance the efficiency and capabilities of cloud workflows further [8]. In cloud computing, workflows play a crucial role in automating and orchestrating complex sequences of tasks to achieve specific objectives efficiently and consistently. These workflows are essential for managing the execution of processes that involve multiple interdependent steps, data transfers, and resource allocations across distributed cloud environments.

Therefore, AI used the data analytics and predictive modelling to enable autonomous decision-making, dynamically optimizing resource allocation by learning from past behaviours and adapting to changing conditions. By continuously monitoring available cloud resources, AI is considered an effective solution for optimizing task allocation. The selection of a scheduling approach depends on factors like task prioritization, deadlines, and optimal resource usage in dynamic, heterogeneous environments. Optimization approaches iterate through fine-tuning parameters to enhance scheduling strategies. Therefore, this paper explores the role of AI-driven optimization in improving resource allocation and workload distribution. The major contributions of the paper are:

- The paper presented an adaptive and hybrid multi-hierarchy workflow scheduling algorithm.
- In the proposed approach a hybridization of task prioritization is done with multi-objective resource allocation approach. Then resource allocation matrix is generated using ensemble machine learning approach.
- Finally, the optimal resource allocation was performed using multi-objective multi-hierarchy approach.
- The result was evaluated with varying number of tasks as well as varying numbers of VMs.



The rest part of the paper is organized as: section 2 presents the recent research contributions for scheduling workflows in cloud architecture. Then in section 3 proposed methodology is described with respective algorithms. Then results are presented in section 4 with comparative analysis. Finally, conclusion and future scope is presented in section 5.

2. LITERATURE REVIEW

Mangalampalli et al. [1] proposed a multi-objective workflow scheduling algorithm using Deep Reinforcement Learning, which maps workflows onto Virtual Machines based on dependencies and electricity costs at datacentres. Simulations demonstrate improved performance, reducing makespan and energy consumption compared to existing methods. Khaleel et al. [2] introduced a weight assignment algorithm for cloud-deployed three-tier applications, focusing on optimizing resource utilization. The CloudSim simulation tool was used to evaluate key performance metrics, including latency, bandwidth, completion time, throughput, energy usage, and QoS violations. Lubal et al. [3] explored the use of Reinforcement Learning (RL) in distributed computing environments to optimize workflow scheduling, focusing on minimizing costs and adapting to dynamic workloads. This approach enhances resource utilization in cloud computing. Yang et al. [4] developed an algorithm that generates workflow instances and applies GA, NSGA, and NSGA-NN optimization methods to find optimal solutions. They also constructed a dataset and trained a seq2seq model for real-time scheduling. Simulation results showed the algorithm's effectiveness, achieving 90% of the optimal scheduling solution. Alsubaei et al. [5] proposed a dual machine learning approach using K-means to optimize cloud scheduling performance. The first method, Efficient K-means, enhances task processing speed and efficiency, while the second, K-means HEFT, improves processing time and efficiency across virtual machines and varying task sizes. Zhou et al. [6] introduced a hybrid machine learning technique for resource allocation and task scheduling, combining IPSO-TS, GANN, and NSUPREME for encryption. Simulations demonstrated that their method outperforms existing approaches in resource utilization, power consumption, and responsiveness.

Samadi et al. [7] proposed an enhanced version of the Heterogeneous Earliest Finish Time (E-HEFT) algorithm, designed to balance the load across virtual machines while minimizing workflow makespan. Despite being NP-complete, the algorithm outperforms existing scheduling methods by reducing makespan and improving load distribution among virtual machines. Kijak et al. [8] explored the scheduling of scientific workflows, introducing the Serverless Deadline-Budget Workflow Scheduling algorithm for serverless platforms. Preliminary experiments using the Montage workflow demonstrated the algorithm's effectiveness in addressing workflow scheduling challenges. Singghal et al. [9] introduced a ranking algorithm based on task rank and predecessors' average communication time, reducing schedule length and improving load balance among resources by creating a task pool. Simulations show that this algorithm outperforms existing methods in load balance, makespan, and turnaround time. Chakravarthi et al. [10] explored resource provisioning and scheduling algorithms in cloud environments, with a focus on Infrastructure as a Service (IaaS). Their work provides a detailed overview of existing techniques and highlights key research



challenges in this area. Choudhary et al. [11] proposed a hybrid workflow scheduling algorithm that combines the Gravitational Search Algorithm (GSA) with Heterogeneous Earliest Finish Time (HEFT), aiming to minimize makespan and cost.

The algorithm, which uses cost-time equivalence and metrics like MCR and SLR, outperforms GSA, HGA, and HEFT in experiments. Jiahao et al. [12] introduced a reinforcement learning-based cloud workflow scheduling algorithm, merging Q-Learning with weighted fitness value functions to reduce execution time and optimize system load, as demonstrated through simulations on real cloud data. Saeedi et al. [13] proposed the MaOPSO algorithm, which focuses on improving workflow scheduling by maximizing reliability while minimizing cost, makespan, and energy consumption. It achieves significant improvements in convergence and HyperVolume criterion compared to existing algorithms like LEAF, MaOPSO, and EMS-C. Malik et al. [14] developed a scheduling algorithm based on task classification and thresholds, employing Particle Swarm Optimization for optimal scheduling. Experiments show that the algorithm effectively improves energy consumption, makespan, and load balancing. Alsurdeh et al. [15] proposed a resource estimation algorithm and a cluster-based provisioning and scheduling technique for hybrid workflows on edge cloud resources. This method optimizes execution time and cost, outperforming meta-heuristic techniques by 8% and 35%, respectively, for large-scale workflows. Mahmoud et al. [16] introduced the Load Balancing HEFT (LB-HEFT) algorithm, which improves load balancing in the HEFT algorithm. Comparative studies show that LB-HEFT outperforms E-HEFT and HEFT by enhancing load balance, resource utilization, and makespan. Garg et al. [17] developed a task scheduling model for virtual machines (VMs) that aims to minimize energy consumption, resource usage, and workflow makespan, while respecting deadlines and dependency constraints. The model incorporates an energy framework, a double-threshold policy, and a live VM migration strategy. Simulations and real-world testing show the ERES algorithm effectively balances workloads and reduces energy consumption.

Tang et al. [18] introduced the Cost-Efficient Task Scheduling Strategy (CETSS), which includes a workflow DDAG model, sub deadline initialization, a greedy scheduling algorithm, and task adjustment. Experimental results show CETSS significantly reduces total workflow execution costs compared to existing algorithms. Maa et al. [19] introduced a real-time multiple-workflow scheduling (RMWS) scheme that dynamically allocates tasks with minimum cost under varying deadline constraints. The algorithm, which operates in three stages i.e., task arrival, allocation, and completion, that outperforms two state-of-the-art methods in terms of rental cost, resource utilization, success rate, and deadline adherence. Ahmad et al. [20] explored scientific workflow management and scheduling in cloud computing, reviewing systems and techniques like resource, fault-tolerant, and energy-efficient scheduling, while discussing key performance metrics, platforms, and research challenges. Mubeen et al. [21] proposed an adaptive load-balanced task scheduling (ALTS) approach that reduces makespan, maximizes resource utilization, and minimizes SLA violations, outperforming existing methods in ARUR, makespan, and SLA violation.

Tang et al. [22] proposed a fault-tolerant, cost-efficient workflow scheduling algorithm (FCWS) that outperforms existing methods like FR-MOS and CWS in terms of cost, reliability, and makespan, based on simulation results. Thekkepurayil et al. [23] introduced a hybrid algorithm combining Ant-Lion Optimization (ALO) and Particle Swarm Optimization (PSO) for cloud workflow scheduling, incorporating Data Encryption Standard (DES) for data encoding. Their study demonstrated that this hybrid approach improves energy consumption, cost efficiency, load balancing, and reliability when compared to existing methods. According to the study presented in above section, it is observed that in cloud computing, factors such as CPU utilization, memory utilization, latency, response time, and speed are critical for virtualization. Virtual machines (VM) load depends on such critical factors and it is required to optimize these factors related with workloads at each hosting servers to avoid congestion as well as to reduce resource wastage. Each VM and host has load bounds and crossing these bounds lowers performance or wastes resources. Load balancing includes migration of workloads from overloaded to underloaded VMs. This combinatorial problem requires optimal VM selection and migration for dynamic workflow loads in cloud architecture.

3. METHODOLOGY USED

The proposed adaptive hybrid multi-hierarchy optimization (AHMHO) algorithm-based workflow scheduling focuses on efficiently scheduling tasks in cloud computing environments by prioritizing tasks and then according to that generation of the resource matrix (RM) with allocation of the optimal execution resources (OER) through a multi-objective optimization process, as presented in fig 1. This methodology combines task prioritization (TP), resource matrix generation (RMG), and multi-hierarchy optimization (MHO) to handle complex cloud workloads.

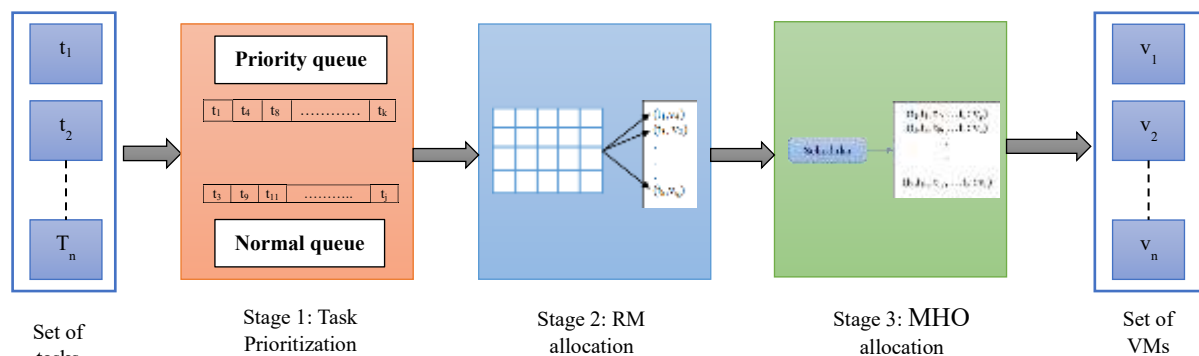


Fig. 1. Flowchart of Adaptive hybrid multi-hierarchy optimization

Task Prioritization

In this stage, each task is evaluated and ranked based on its importance, characteristics, and execution requirements. The objective is to ensure that high-priority tasks are handled with the required resources while maintaining overall system efficiency. For this task ranking is performed as:

- Each task is analysed based on features like computational complexity, latency sensitivity, resource requirements, and deadlines.

- Tasks are ranked using an adaptive prioritization mechanism. A set of ranking criteria (e.g., task size, urgency, and dependency) is applied to order tasks. Each task T_i is assigned a priority score P_i .

$$P_i = \sum_{k=1}^m w_k * f_k(T_i) \quad (1)$$

Where, w_k is the weight assigned to feature f_k of task T_i . The task ranking mechanism can adapt based on dynamic factors such as real-time system load and changing task characteristics.

Resource Matrix (RM) Generation

Once tasks are prioritized, the next step is generating the Resource Matrix (RM). This matrix maps available cloud resources i.e., VMs to tasks based on their respective characteristics and resource availability. An ensemble learning approach is used to predict the availability and suitability of resources for each task. The input features for each task include resource requirements, computational complexity, and dependency relationships. The collected features are: Task priority, task size, task cost, and assigned VM. These features are used generate a feature matrix as $F_m = \{f_1, f_2, \dots, f_n\}$. Then ensemble model is used to train according to the presented feature vector generated out of historical data. The ensemble model predicts the availability matrix as $A_{ij} = f_{ensemble}(F_m)$. Then these availability matrix is used to allocate available resources.

Resource Allocation

The Multi-Hierarchy Optimization (MHO) is an extension of the Particle Swarm Optimization (PSO) algorithm, where particles are organized into hierarchical levels. This method combines both global exploration (higher hierarchy) and local exploitation (lower hierarchy) to optimize a multi-objective scheduling problem by minimizing makespan and energy consumption while balancing workload across VMs. The scheduling problem is modelled as a multi-objective optimization problem with the following goals:

- Minimizing the makespan of tasks across VMs.
- Optimizing energy consumption for the execution of tasks.
- Minimizing load imbalance across VMs.
- Minimizing response time and average delay time for tasks.

In this setup, we have N tasks and M VMs. Each task must be assigned to a VM, and each VM has varying execution times and energy consumption for different tasks. Each particle X_i represents a vector of task priorities for task assignment:

$$X_i = [p_1, p_2, \dots, p_n] \quad (2)$$

Where p_i is the priority of task i .

The velocity v_i for each particle is updated using the following equation:

$$V_i^{t+1} = w * V_i^{(t)} + c_1 r_1 (P_{best,i} - X_i^{(t)}) + c_2 r_2 (G_{best} - X_i^{(t)}) \quad (3)$$

Where, the velocity of particle i at iteration t is represented as $V_i^{(t)}$, $P_{best,i}$ represents the personal best solution around i and the global best position was G_{best} with cognitive and social

parameters as c_1 and c_2 respectively. w is the inertia weight that will control the influencing parameter of the previous velocity.

The position of the X_i is updated as:

$$X_i^{(t+1)} = X_i^{(t)} + V_i^{(t+1)} \quad (4)$$

The final objective function used here is:

$$f(X_i) = Makespan(X_i) + \lambda_1 * DI(X_i) + \lambda_2 Energy(X_i) \quad (5)$$

Where, λ_1 and λ_2 are weights.

MHO takes advantage of unused time slots on VMs by scheduling tasks in gaps between already scheduled tasks. This improves resource utilization. MHO treats the problem as a multi-objective optimization where conflicting goals (e.g., minimizing makespan vs. minimizing energy consumption) are balanced by using weights λ_1 and λ_2 in the objective function. The algorithm iteratively updates particle positions based on the objective function and converges to an optimal task schedule. Each task is assigned a priority based on its characteristics, such as latency, average delay, etc. which helps determine the order of execution. Tasks are allocated to VMs based on their execution time and resource requirements. The proposed method optimizes this process by minimizing makespan and balancing the load across VMs.

Algorithm

Input:

Set of tasks $T = \{T_1, T_2, \dots, T_n\}$ with task features FT_i .

Set of virtual machines $VM = \{VM_1, VM_2, \dots, VM_n\}$ with resource capabilities

Objective weights λ_1, λ_2 for balancing load and energy consumption.

Output:

Optimal resource allocation $X_i^{(t+1)}$ for each task.

Begin

For each task T_i in T such that $T_i \in T$:

Calculate the priority score P_i .

Sort the tasks T based on the P_i .

For each task $T_i \in T$ and each VM $VM_j \in VM$:

Evaluate $RM[i, j] = f_{ensemble}(T_i, VM_j)$

Evaluate $f(X_i) = Makespan(X_i) + \lambda_1 * DI(X_i) + \lambda_2 Energy(X_i)$

End for

End for

Return $X_i^{(t+1)}$

4. RESULTS AND DISCUSSION

In this section, the performance of workflow scheduling and cloud load balancing is evaluated using the MATLAB simulation tool that is widely used for workflow scheduling simulation. MATLAB is used for designing the simulation modelling and experimentation. The evaluation

is conducted on a 64-bit Windows system with an Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz (2.11 GHz) and 8GB RAM. For this following parameter are used:

Makespan Time (MT): MT in cloud workflow scheduling is the total time taken to complete a set of tasks.

$$MT = \max\{exe_{time_i}\} \quad where, 0 \leq i \leq n \quad (6)$$

Average Delay Time (ADT): For execution of n tasks, the average time consumed is evaluated by ADT.

$$ADT = \frac{\sum_{i=1}^n exe_{time_i}}{n} \quad (7)$$

Degree of Imbalance (DI): It is represented as the difference between max and min response time with respect to ADT.

$$DI = \frac{Max(R_i) - Min(R_i)}{ADT} \quad (8)$$

Standard Deviation (SD): It is evaluated for estimation of distribution of load over the servers.

$$SD = \sqrt{\frac{\sum (R_i - ADT)^2}{n}} \quad (9)$$

The fig 2 illustrates a comparison between three task scheduling approaches i.e., Priority Based, PSO Priority Based, and Proposed method with respect to makespan time with increasing number of tasks. While all methods perform similarly for a small number of tasks but the difference becomes evident with larger workloads. Both the Priority Based and PSO Priority Based approaches show a significant increase in makespan time as the task count reaches 100 and 500, with PSO being marginally more efficient. However, the Proposed method demonstrates a lower makespan time. This suggests that the Proposed method is more efficient and scalable as it better for handling large task sets in cloud workflow scheduling.

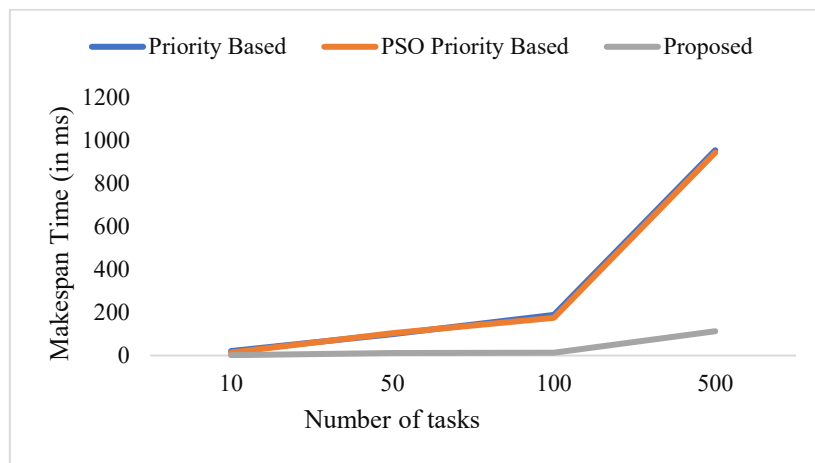


Fig. 2. Makespan Time Evaluation with respect to Workflow Tasks

The fig 3 presents the Average Delay Time (ADT) in ms for three different task scheduling approaches i.e., Priority Based, PSO Priority Based, and Proposed with varying workload size from 10-500. For smaller workloads all three methods perform similar with very low average delay times.

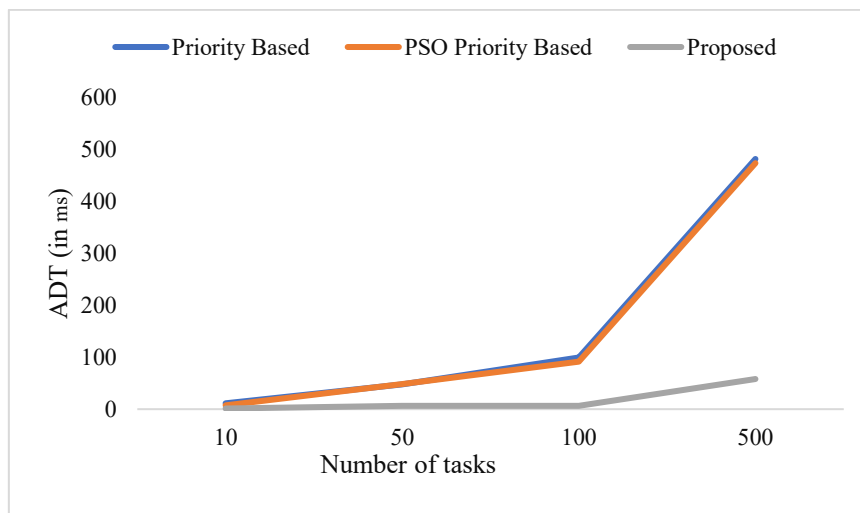


Fig. 3. ADT Evaluation with respect to Workflow Tasks

As the number of tasks increases the Priority Based and PSO Priority Based approaches experience an increase in ADT. The Proposed method maintains consistently lower ADT across all task sizes. Therefore, the Proposed method is clearly more efficient in reducing ADT as compared to others. As the task count increases the Proposed method shows its strength in maintaining lower delays under heavy workloads.

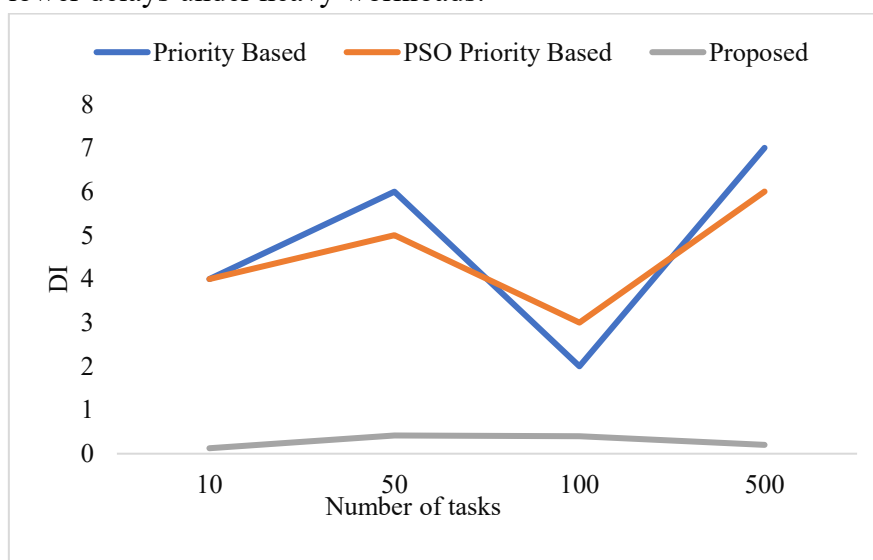


Fig. 4. DI Evaluation with respect to Workflow Tasks

The fig 4 presents the DI for three different approaches i.e., priority Based, PSO Priority Based, and Proposed. The Priority Based and PSO Priority Based approaches show fluctuations in the DI as the number of tasks increases. The Priority Based approach has the highest DI. The PSO Priority Based approach follows a similar trend but with slightly lower DI values compared to the Priority Based approach across all task sizes. Whereas the Proposed approach shows a very stable and low DI across all task sizes.

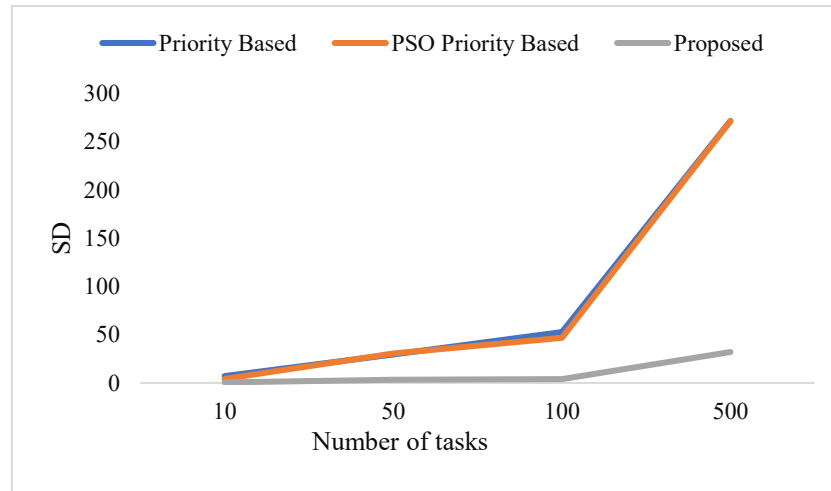


Fig. 5. SD Evaluation with respect to Workflow Tasks

The fig 5 presents the SD of the scheduling performance for three different approaches. For smaller task sizes, the Priority Based, PSO Priority Based, and Proposed approaches all maintain low SD. As the number of tasks increases, the Priority Based and PSO Priority Based methods show a steep rise in SD that reflects is inconsistency in performance as task complexity grows. The sharp rise in SD for the Priority Based and PSO Priority Based methods at higher task numbers suggests these approaches become less stable. Whereas the proposed approach demonstrates much better stability and consistency in scheduling performance. Lower SD in the proposed method indicates that it can maintain more reliable and predictable outcomes in scheduling that makes it better for environments requiring consistent task execution under heavy workloads.

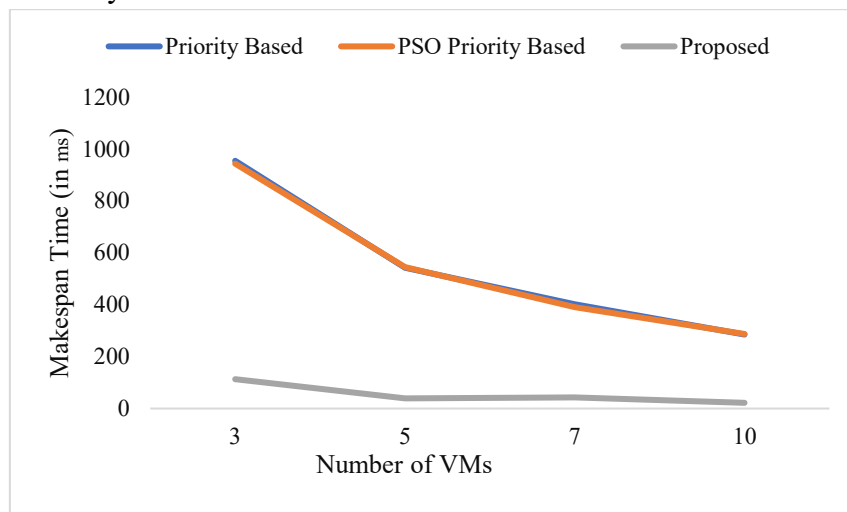


Fig. 6. Makespan Time Evaluation with respect to VMs

The fig 6 compares the Makespan Time across three scheduling approaches i.e., Priority Based, PSO Priority Based, and the Proposed method with varying numbers of Virtual Machines (VMs). As the number of VMs increases, all approaches show a reduction in makespan time. However, the Proposed method outperforms better as compared to both Priority Based and PSO Priority Based. While the Priority Based and PSO Priority Based methods improve as the

number of VMs increases. Whereas the Proposed method remains highly efficient that shows that it is better in optimization and resource utilization.

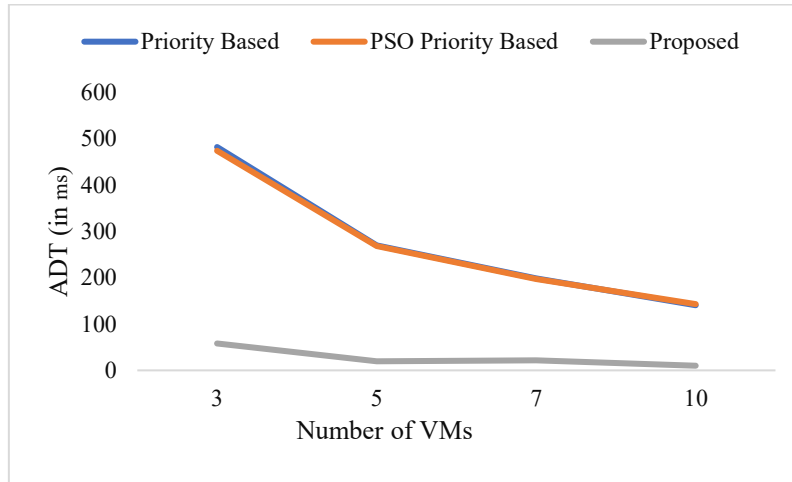


Fig. 7. ADT Evaluation with respect to VMs

The fig 7 compares the ADT for three scheduling approaches i.e., Priority Based, PSO Priority Based, and Proposed with varying numbers of VMs. Both the Priority Based and PSO Priority Based approaches shows a decrease in ADT as the number of VMs increases. However, the Proposed method consistently maintains a lower ADT across all VM counts, showing minimal variation even as the number of VMs grows. This demonstrates that the Proposed approach is more efficient and scalable that is more efficient in minimizing delays even with fewer resources.

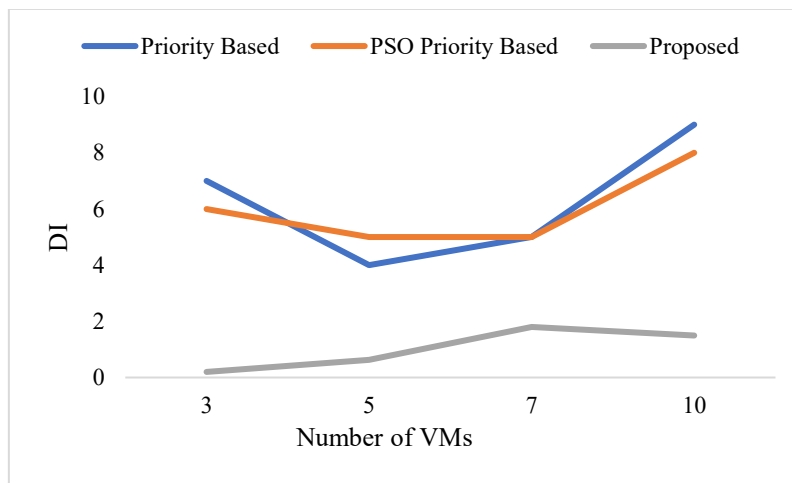


Fig. 8. DI Evaluation with respect to VMs

The fig 8 compares the DI for three scheduling approaches i.e., Priority Based, PSO Priority Based, and Proposed with varying VM. Both the Priority Based and PSO Priority Based approaches show higher DI values that shows high inconsistency in task scheduling. Whereas, the proposed method maintains a consistently lower DI that shows a more uniform and stable task distribution. This suggests that the Proposed approach is more reliable and efficient with increasing number of VMs.

The fig 9 compares the SD of Priority Based, PSO Priority Based, and Proposed with varying numbers of VMs. The Priority based and PSO Priority based methods show high variability in performance with fewer VMs but their performance improves as the number of VMs increases. Whereas the proposed method maintains low SD values across all VM counts showing it as more reliable as compared to others.

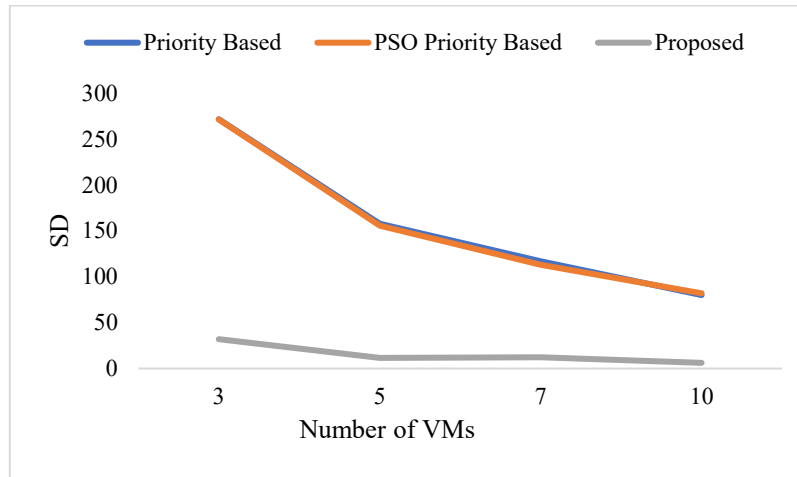


Fig. 9. SD Evaluation with respect to VMs

5. CONCLUSION

This paper presents a comprehensive evaluation of cloud workflow scheduling using adaptive hybrid Multi-Hierarchy Optimization (AHMHO). The simulation results indicate that the proposed method outperforms Priority Based and PSO Priority Based approaches in terms of makespan time, ADT, DI, and SD with a large number of tasks or Virtual Machines (VMs). The proposed method demonstrates superior scalability and resource optimization as the workload size and the number of VMs increase. The AHMHO approach consistently maintains lower makespan and ADT, stable DI, and reduced SD across different task sizes and VM counts. The efficient handling of resource utilization and better load balancing under heavy workloads further highlights the proposed methodology capability to meet the growing demands of cloud computing environments. In future this work will be extended to workload type-based scheduling strategy.

6. REFERENCES

- [1] S. Mangalampalli *et al.*, “Multi Objective Prioritized Workflow Scheduling Using Deep Reinforcement Based Learning in Cloud Computing,” *IEEE Access*, vol. 12, pp. 5373–5392, 2024, doi: 10.1109/ACCESS.2024.3350741.
- [2] M. I. Khaleel, “A dynamic weight–assignment load balancing approach for workflow scheduling in edge-cloud computing using ameliorated moth flame and rock hyrax optimization algorithms,” *Futur. Gener. Comput. Syst.*, vol. 155, pp. 465–485, 2024, doi: <https://doi.org/10.1016/j.future.2024.02.025>.
- [3] Lubal, Omkar Durva. Scientific Workflow Scheduling using Reinforcement Learning. Diss. CALIFORNIA STATE UNIVERSITY, NORTHRIDGE, 2024.



- [4] Z. Yang, M. Zhang, H. Li, and W. Ding, "A novel seq2seq-based prediction approach for workflow scheduling," *Cluster Comput.*, vol. 27, no. 2, pp. 1897–1910, 2024, doi: 10.1007/s10586-023-04061-3.
- [5] F. S. Alsubaei, A. Y. Hamed, M. R. Hassan, M. Mohery, and M. K. Elnahary, "Machine learning approach to optimal task scheduling in cloud communication," *Alexandria Eng. J.*, vol. 89, pp. 1–30, 2024, doi: <https://doi.org/10.1016/j.aej.2024.01.040>.
- [6] H. Zhou, "A Novel Approach to Cloud Resource Management: Hybrid Machine Learning and Task Scheduling," *J. Grid Comput.*, vol. 21, no. 4, p. 68, 2023, doi: 10.1007/s10723-023-09702-w.
- [7] Y. Samadi, M. Zbakh, and C. Tadonki, "E-HEFT: Enhancement Heterogeneous Earliest Finish Time algorithm for Task Scheduling based on Load Balancing in Cloud Computing," in *2018 International Conference on High Performance Computing & Simulation (HPCS)*, 2018, pp. 601–609. doi: 10.1109/HPCS.2018.00100.
- [8] J. Kijak, P. Martyna, M. Pawlik, B. Balis, and M. Malawski, "Challenges for Scheduling Scientific Workflows on Cloud Functions," in *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*, 2018, pp. 460–467. doi: 10.1109/CLOUD.2018.00065.
- [9] S. Singhal and J. Patel, "Load balancing scheduling algorithm for concurrent workflow," *Comput. Informatics*, vol. 37, no. 2, pp. 311–326, 2018, doi:10.4149/cai_2018_2_311.
- [10] Chakravarthi, D. Mlakić, and E. Alibašić, "Thermal and arc flash analysis of electromotive drives in industrial networks," *Int. J. Electr. Comput. Eng.*, vol. 8, no. 2, pp. 853–866, 2018, doi: 10.11591/ijece.v8i2.pp853-866.
- [11] A. Choudhary, I. Gupta, V. Singh, and P. K. Jana, "A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing," *Futur. Gener. Comput. Syst.*, vol. 83, pp. 14–26, 2018, doi: <https://doi.org/10.1016/j.future.2018.01.005>.
- [12] W. Jiahao, P. Zhiping, C. Delong, L. Qirui, and H. Jieguang, "A Multi-object Optimization Cloud Workflow Scheduling Algorithm Based on Reinforcement Learning," in *Intelligent Computing Theories and Application*, 2018, pp. 550–559.
- [13] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanzpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," *Comput. Ind. Eng.*, vol. 147, p. 106649, 2020, doi: <https://doi.org/10.1016/j.cie.2020.106649>.
- [14] N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, "Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds," *Appl. Sci.*, vol. 11, no. 13, 2021, doi: 10.3390/app11135849.
- [15] R. Alsurdeh, R. N. Calheiros, K. M. Matawie, and B. Javadi, "Hybrid Workflow Scheduling on Edge Cloud Computing Systems," *IEEE Access*, vol. 9, pp. 134783–134799, 2021, doi: 10.1109/ACCESS.2021.3116716.
- [16] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "An efficient load balancing technique for task scheduling in heterogeneous cloud environment," *Cluster Comput.*, vol. 24, no. 4, pp. 3405–3419, 2021, doi: 10.1007/s10586-021-03334-z.



- [17] N. Garg, D. Singh, and M. S. Goraya, "Energy and resource efficient workflow scheduling in a virtualized cloud environment," *Cluster Comput.*, vol. 24, no. 2, pp. 767–797, 2021, doi: 10.1007/s10586-020-03149-4.
- [18] X. Tang et al., "Cost-Efficient Workflow Scheduling Algorithm for Applications With Deadline Constraint on Heterogeneous Clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 9, pp. 2079–2092, 2022, doi: 10.1109/TPDS.2021.3134247.
- [19] X. Ma, H. Xu, H. Gao, and M. Bian, "Real-Time Multiple-Workflow Scheduling in Cloud Environments," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 4, pp. 4002–4018, 2021, doi: 10.1109/TNSM.2021.3125395.
- [20] Z. Ahmad et al., "Scientific Workflows Management and Scheduling in CloudComputing: Taxonomy, Prospects, and Challenges," *IEEE Access*, vol. 9, pp. 53491–53508, 2021, doi: 10.1109/ACCESS.2021.3070785.
- [21] A. Mubeen, M. Ibrahim, N. Bibi, M. Baz, H. Hamam, and O. Cheikhrouhou, "Alts: An Adaptive Load Balanced Task Scheduling Approach for Cloud Computing," *Processes*, vol. 9, no. 9, 2021, doi:10.3390/pr9091514.
- [22] X. Tang, "Reliability-Aware Cost-Efficient Scientific Workflows Scheduling Strategy on Multi-Cloud Systems," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2909–2919, 2022, doi: 10.1109/TCC.2021.3057422.
- [23] J. Kakkottakath Valappil Thekkepurayil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Comput.*, vol. 24, no. 3, pp. 2367–2384, 2021, doi: 10.1007/s10586-021-03269-5.