IJEMD-CSAI, 5 (1) (2026) https://doi.org/ 10.54938/ijemdcsai.2026.05.1.603

International Journal of Emerging Multidisciplinaries:
Computer Science and Artificial Intelligence

Research Paper
Journal Homepage: www.ojs.ijemd.com
ISSN (print): 2791-0164 ISSN (online): 2957-5036

A Two Stage Deadline Aware Workflow Scheduling Strategy for Load Balancing in Cloud Environment

Aminu Ibrahim Inuwa[1], Faruk Umar Ambursa[2], Muhammad Yusuf Muhammad[3] and Aliyu Shuaibu[4]

1. Department of Computer Science, Faculty of Computing, Bayero University Kano, Nigeria
2. Department of Information Technology, Faculty of Computing, Bayero University Kano, Nigeria
3. Department of Computer Science, Faculty of Computing, Bayero University Kano, Nigeria
4. Department of Computer Science, Faculty of Computing, Bayero University Kano, Nigeria

Abstract

Cloud computing enables on-demand access to shared computational resources, but effective load balancing remains a major challenge in heterogeneous environments. Scheduling must be performed quickly while optimizing resource utilization and meeting workflow deadlines, a task complicated by job dependencies, dynamic workloads, and variable execution times. The commonly used Max-Min scheduling algorithm assigns larger tasks to faster resources; however, it often produces a high makespan and poor resource utilization when long tasks dominate the workload. To address this limitation, this study proposes a two-stage deadline-aware workflow scheduling strategy designed to reduce makespan and prevent deadline violations. In the first stage, tasks are prioritized based on deadline constraints and dependency levels to ensure time-critical tasks are scheduled earlier, while in the second stage, resource allocation is optimized using execution time estimation to balance workload distribution across available resources. The proposed method was evaluated across twelve scenarios involving three different workflow types and compared with a benchmark approach. Experimental results show that the proposed strategy consistently achieves a reduced makespan compared to the conventional Max-Min algorithm, demonstrating improved efficiency and performance in cloud computing environments.

Keywords- *Cloud Environment, Cloud computing Max-Min algorithm, two-stage deadline-aware, scheduling,*

Email Address: aliyu2017@gmail.com (Aliyu Shuaibu[4])

INTRODUCTION

Cloud computing provides on-demand access to shared computational resources, enabling organizations and individuals to utilize scalable infrastructure without the need for significant upfront investment in hardware and software [1]. Through virtualization, multiple users can share physical resources efficiently, improving flexibility and supporting dynamic load balancing [2]. Cloud systems are typically categorized based on deployment models (e.g., public, private, hybrid) and service models such as Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

As the number of cloud users and applications increases, efficient task scheduling becomes a critical challenge. Task scheduling refers to the process of assigning tasks to available resources over time to ensure optimal execution [5]. In cloud environments, user-submitted tasks are processed in data centers, where resources must be allocated and scheduled effectively to meet performance requirements. Since scheduling directly influences system efficiency, it plays a key role in determining the reliability and overall performance of cloud services [2].

Task scheduling in distributed cloud environments is an NP-hard problem due to the complexity of managing heterogeneous resources, task dependencies, and dynamic workloads [5]. Consequently, designing efficient scheduling algorithms that minimize makespan while maximizing resource utilization remains an active research area. Various scheduling strategies have been proposed to improve system performance, considering key constraints such as execution time, cost, throughput, and resource utilization [10], [19]. Among these, makespan minimization and efficient resource usage are particularly important, as they directly impact user satisfaction and service provider efficiency.

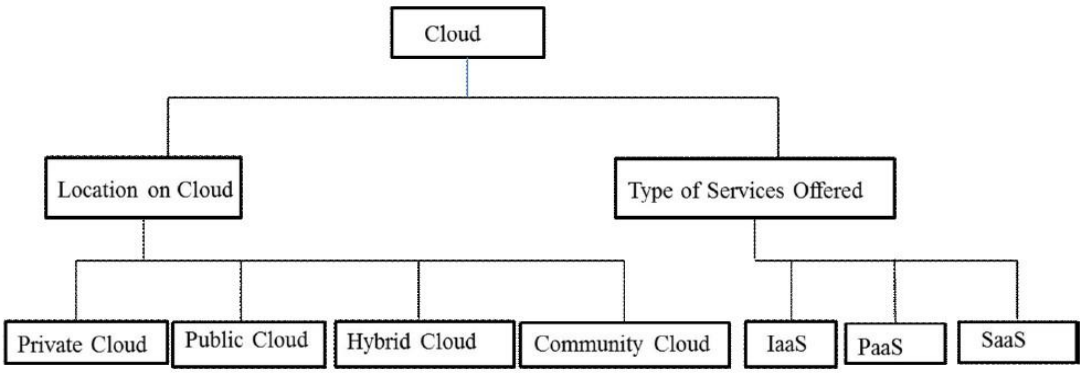

Figure 1: Cloud Platform Adopted from [6]

i. Challenges of Cloud Computing

Cloud computing provides scalable and flexible services; however, it presents several evolving research challenges. One major issue is managing dynamic and unpredictable workloads, where resource demand fluctuates due to changing user activities and application requirements. These variations

make it difficult to maintain efficient resource allocation and consistent performance [7]. In addition, ensuring optimal resource utilization in heterogeneous cloud environments comprising nodes with different processing speeds, memory capacities, and network capabilities requires intelligent scheduling mechanisms capable of adapting to real-time system conditions. Effective resource management also involves virtual machine migration and server consolidation to maintain load balance, reduce energy consumption, and improve system reliability [8]. While these techniques enhance flexibility and cost efficiency, they introduce challenges such as migration overhead, service disruption, and the risk of overloading consolidated servers. Furthermore, maintaining data security and enforcing quality of service (QoS) requirements remain critical concerns, particularly in large-scale distributed infrastructures where resources are shared and dynamically allocated.

Given these challenges, rapid and efficient job assignment in heterogeneous cloud environments is essential. Traditional scheduling approaches, such as Max-Min-based strategies, often fail to simultaneously minimize makespan, maximize resource utilization, and ensure deadline adherence. Therefore, there is a need for a robust, multi-objective workflow scheduling framework. To address this gap, this study proposes a two-stage, deadline-aware scheduling strategy that integrates task prioritization with best-fit resource allocation, providing a more adaptive and efficient solution for workflow execution in dynamic cloud computing environments.

RELATED WORKS

This section reviews existing research on scheduling heuristics based on the Max-Min approach. The discussion highlights their methodologies, advantages, and inherent limitations, with a focus on how each work contributes to the evolution of Max-Min–based scheduling in distributed and cloud computing environments.

The evolution of hybrid methods gained further traction. [4] introduced a Hybrid Max-Min Genetic Algorithm (HMMGA) that integrates Max-Min scheduling with the global optimization capabilities of Genetic Algorithms (GA). Unlike traditional Max-Min, which risks overloading resources by always assigning the longest tasks first, HMMGA incorporates genetic operations selection, crossover, and mutation to refine task allocation iteratively. Their findings revealed significant improvements in performance**:** a reduction in makespan by 1.6 – 3.8 seconds**,** waiting time reductions of up to 25 seconds**,** and resource utilization gains of 10 – 40% compared to classical Max-Min and other metaheuristic methods such as TOPSIS-PSO. Notably, HMMGA also minimized the degree of imbalance, making it more effective in evenly distributing workloads across heterogeneous cloud environments. Similarly, [3] proposed an enhanced workflow scheduling algorithm, Max-Min++**,** to address the limitations of the traditional Max-Min approach in cloud environments. Their method integrates an Iterative Mean-Sort with Fair Selection Technique that balances the allocation of short and long tasks by considering average task length during scheduling. Implemented in WorkflowSim and evaluated using Montag**e** and Cybershake

scientific workflows, the algorithm consistently outperformed the benchmark Max-Min+ by significantly reducing makespan. The study demonstrates that Max-Min++ achieves better task distribution and fairness, though its evaluation was limited to makespan, leaving room for future research on other performance metrics such as cost, throughput, and energy efficiency

[16] proposed RDLBS2, a receiver-initiated deadline-aware load balancing strategy designed to reduce task completion delays in cloud environments by migrating cloudlets to virtual machines with sufficient remaining capacity. Evaluated using CloudSim, the method improved performance especially turnaround time compared to existing load balancing techniques. However, the approach mainly focuses on VM-level migration and does not address workflow-level scheduling or a structured multi-stage framework, highlighting the need for more adaptive, comprehensive deadline-aware scheduling strategies.

[18] reviewed cloud resource scheduling techniques, analyzing heuristic, meta-heuristic, and hybrid approaches based on objectives such as cost, makespan, load balancing, energy efficiency, reliability, security, and SLA compliance. The study highlighted strengths, limitations, and open research challenges in current methods but remained a theoretical review without proposing a concrete scheduling framework for dynamic, heterogeneous cloud environments.

Similarly, [19] proposed a priority-based load balancing (PLB) approach that uses a multi-queue architecture to prevent task starvation and improve resource utilization in cloud environments. Implemented in CloudSim 3 and evaluated against several traditional algorithms, PLB showed improved makespan, response time, and resource efficiency. However, the method mainly emphasizes priority handling and queue management, without incorporating explicit deadline awareness or a comprehensive multi-stage scheduling framework.
[9] presented PSO-CALBA, a hybrid content-aware load balancing technique that combines Particle Swarm Optimization for effective task-to-resource mapping with a Support Vector Machine classifier for task classification. The approach outperformed other methods in terms of makespan and workload balancing when tested in CloudSim. Nevertheless, it does not specifically handle workflow dependencies or time limitations, and thus adds extra computational expense.

Furthermore, [2] proposed a dynamic task scheduling model based on a Stackelberg game to manage budget and deadline constraints in cloud environments. By incorporating pricing, satisfaction factors, and resource utilization, the approach achieved improvements in makespan, deadline adherence, cost, throughput, and provider profit compared to several existing algorithms. However, its game-theoretic design introduces high computational complexity, which may limit scalability and real-time application in highly dynamic cloud systems.

In another work, [7] proposed RDLBS2, a receiver-initiated deadline-aware load balancing strategy that dynamically migrates cloudlets to suitable virtual machines based on remaining processing capacity to improve deadline compliance and reduce turnaround time. Evaluated in CloudSim, the approach showed better task completion and resource utilization than existing methods. However, it mainly targets VM-level load balancing and lacks multi-stage workflow scheduling and adaptive mechanisms for highly dynamic cloud environments.

Moreover, [10] introduced a Stackelberg game–based dynamic task scheduling model to handle budget and deadline constraints in cloud environments. By modeling interactions among tasks, schedulers, and resources using pricing and QoS factors, the approach achieved notable improvements in makespan, deadline compliance, cost reduction, throughput, and provider profit compared to several existing methods. However, the reliance on dynamic game-theoretic computations introduces additional complexity that may limit scalability in highly dynamic cloud systems.

Highlighted growing interest in cost-aware scheduling within cloud fog environments, while [8] proposed the COTD algorithm, which optimizes operational cost by considering task deadlines when selecting data centers. Evaluated with Cloud Analyst, COTD reduced overall cost while maintaining response time performance compared to traditional routing methods. However, the approach mainly focuses on routing decisions and does not fully address workflow-level or multi-objective scheduling in heterogeneous cloud environments.

METHODOLOGY

i. Workflow Scheduling Problem

Workflow applications can be modelled as a Directed Acyclic Graph (DAG), G = (T, E) where T is a collection of tasks = {T1, T2, T3, …..,Tn} and E is a collection of edges where each edge connects each two tasks and denotes their data dependency (Rodriguez M and Buyya R). Each task in a workflow can only perform in cases where all its parents have been completely processed. This kind of problem is considered an NP-Complete which means no algorithm can generate its solution within polynomial time period. In this work one of the heuristic algorithm that can generate the solution in polynomial time is studied.

The Max-Min algorithm is a heuristic scheduling method that assigns tasks to virtual machines (VMs) based on expected completion times, typically allocating the largest task to the fastest resource and proceeding in descending order of task size [13]. While simple, this approach can lead to high makespan and poor resource utilization, especially when there are many long tasks, as shorter tasks may occupy faster resources and longer tasks slower ones. To overcome these limitations, the Extended Max-Min algorithm improves performance by first calculating completion times for cloudlets and VMs, then sorting cloudlets into maximum and minimum queues, and finally assigning tasks from each queue to the resource that minimizes completion time. This method achieves better load balancing and reduces computational cost compared to the original Max-Min approach.

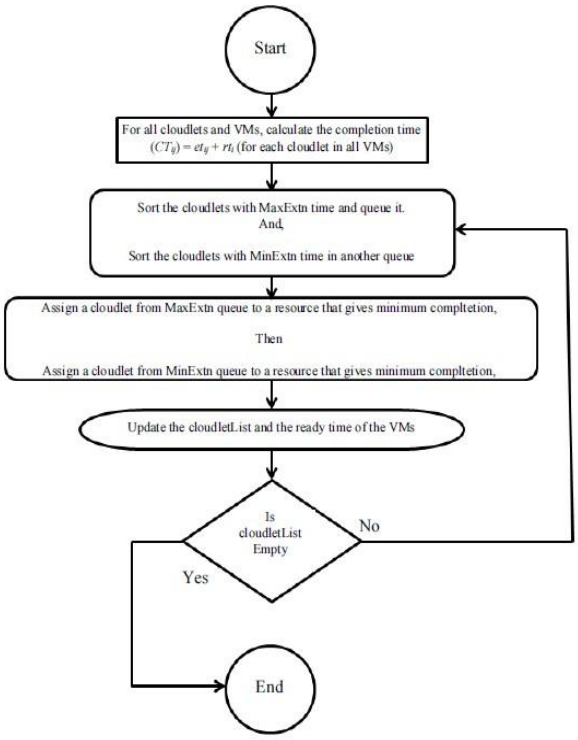

Figure 2: Max-min Algorithm

ii. The Proposed Solution

- The proposed workflow scheduling. We will take a two-stage approach to solve the problem. In the first stage, Tasks will be ordered by their scheduling priorities which are based upon upward and downward ranking according to equation 1 and 2. Rank values reflect the expected performance of a particular task on a particular resource. For all the tasks and virtual machines, the completion time will be calculated and the tasks will be sorted into maximum and minimum execution time.
- The mean of the tasks in the workflow will be calculated.

As a result of this step, we will associate rank values for a set of resources for a particular task.

$$rank_u(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \left(\overline{c_{i,j}} + rank_u(n_j)\right) \quad (1)$$

$$rank_d(n_i) = \max_{n_j \in pred(n_i)} \left\{rank_d(n_j) + \overline{w_j} + \overline{c_{j,i}}\right\} \quad (2)$$

In the second stage, we use our proposed heuristic to obtain a mapping of tasks to resources in the following scenario;

- If the task is greater than or equals to the mean, the task from the maximum queue with minimum execution will be assigned to the resource that best-fits the resource which gives minimum completion.
- If the task is less than, the task from the minimum queue with minimum execution will be assigned to the resource that best-fits the resource which gives minimum completion.
- Remove the assigned tasks and update the ready time of the VM.
- The algorithm ends if there are no tasks in the Meta-task else it restarts again.

iii. Best-Fit

From previous researches, tasks are allocated to resources without considering the amount of time that tasks take before they finish execution. Therefore this underutilization of resources must be addressed. In our proposed model, we introduce the best-fit algorithm.

The best fit deals with allocating the smallest free space in VM which meets the requirement of the requesting task. It will first search the entire list of free space in the VM and considers the smallest space that is adequate. It then tries to find a space which is close to actual task size needed.

Best-fit will allocate a task to the resource that best-fit the resource thereby utilizing the resources.

1. Initialize the resources
   $Lj = 0$ for all $rj \in R$
2. For each task **t**i $\in$ T
   - Find the resource rj such that:

     $Cj - Lj \geq si$ and $(Cj - Lj - si)$ is minimized
   - If such a resource is found, assign ti to rj and update the load

     $Lj = Lj + si$
3. Repeat until all tasks are assigned**.**

iv. Makespan

It is the overall amount of time needed to arrange a cloudlet. This is mostly used to gauge how effective scheduling algorithms are in terms of time. Equations 1 and 2 are used to measure it.

Makespan = max(CT of j(ti, mj)) Equation 1

CT = the job end time – the job start time Equation 2

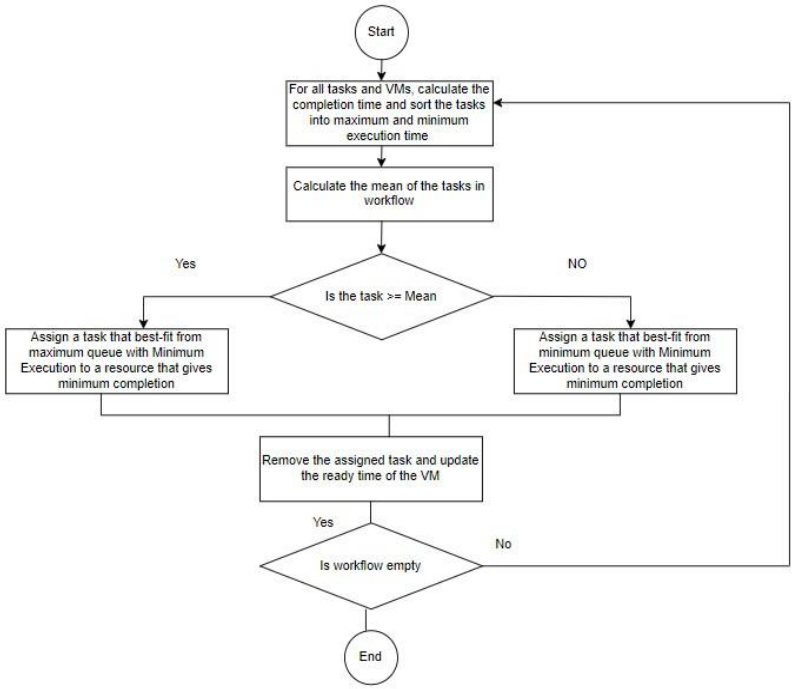

Figure 3: Proposed Model

v. Performance Metrics Evaluation

This study aims to enhance resource utilisation, makespan (ms), and load balancing. Three crucial parameters for assessing a task scheduling algorithm in cloud computing are load balancing, makespan (ms), and resource utilisation.

In workflow scheduling research, it is common to represent the set of tasks as $T_i = \{T_1,T_2,T_3,...,T_n\} \forall i = \{1,2,3,...,n\}$(1) and the set of virtual machines as VM={VM1,VM2,...,VMm}VM = \{VM_1, VM_2, ..., VM_m\}VM={VM1,VM2,...,VMm} (2) where each task is assigned to a VM for execution [15]. The assignment is typically non-preemptive, meaning once a task is allocated to a VM, it runs to completion without interruption. This modeling approach allows for clear mathematical formulation and facilitates the evaluation of scheduling algorithms under various constraints and objectives. In many studies, it is also assumed that VMs have homogeneous capacities to simplify analysis [16].

The set of tasks varying from 1 to n, where all assign to VMs using nonpreemptive approach. The set of VMs is varying from 1 to m. Whereas all the VMs have the same capacity. The execution time represents with ET of any task is mathematically defined in Equation 3. Where FT indicates finish time of task ´ı ´ın VM ´j a´nd ST represents start time of task ´ı ´ın VM ´j ´receptively.

$$ET(ij) = FTij - STij \quad (3)$$

Sum of execution time of task T is represented in Equation 4:

$$ET_{total} = \sum_{i=1}^{n} ET_{ij} \quad (4)$$

After calculating the total execution of tasks, Task which have maximum execution time is selected for resource allocation on VM. Equation5 represents the VM task allocation status.

$$VMs_j = \begin{cases} \\ \end{cases} \quad 1, \text{ if VM is free.} \qquad (5)$$

vi. Standard Workflows

Workflows, originally developed as business process modeling tools to automate and optimize organizational activities [18], have been widely adopted in scientific domains to manage complex, large-scale computational processes. Scientific workflows consist of interconnected computational tasks and are used to conduct experiments, analyze data, and test hypotheses in distributed environments. Examples include the I/O-intensive Montage workflow, which creates sky mosaics by reprojecting, standardizing, and merging astronomical images; the data- and memory-intensive Cybershake workflow for earthquake hazard characterization; LIGO for detecting gravitational waves; SIPHT for automating small RNA gene searches in bacterial genomes; and Epigenomics, a CPU-intensive workflow for genome sequencing operations. These workflows demonstrate the diverse computational and data requirements across scientific applications, highlighting their role in efficiently executing complex, resource-demanding tasks.

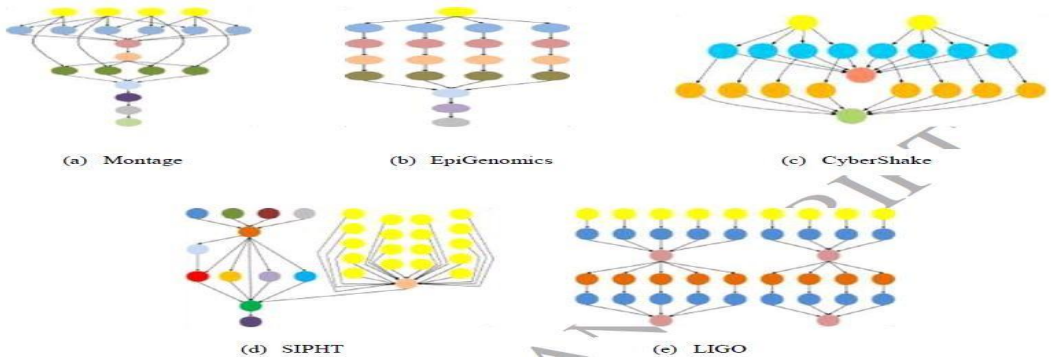

Figure 4: Structure of 5 different scientific workflows: Montage (Astronomy), Cybershake (Earthquake science), Epigenomics (Bioinformatics), SIPHT (Bioinformatics), and LIGO (astrophysics)

RESULT AND DISCUSSION

This chapter presents the results obtained from an experiment on the workflowsim, along with a detailed discussion and a comparison with a benchmark study. A thorough comprehension of the scheduling algorithm's performance is provided by the logical presentation of the results. Overall, this chapter is a valuable resource for researchers and practitioners interested in the deadline aware scheduling algorithm, providing a comprehensive analysis.

i. Result presentation

In this subsection, we present the outcomes of the simulation experiment, with the configurations involving 1 datacenter, ten (10) virtual machines which involves three (3) workflow distribution

Consequently, we evaluated a total of twelve (12) scenarios (4 per each workflow) during the experiments.

ii. Sight Workflow

The suggested scheduling strategy's performance under various workloads was first assessed using the Sight workflow. This process is very helpful for evaluating scheduling algorithms' effectiveness while handling different task complexity levels. Figure 4.1 and Table 4.1 display the makespan achieved by the recommended approach compared to the benchmark research.

Table 1: Makespan of Sight

	Sight light (ms)	Sight Median (ms)	Sight large (ms)	Sight Heavy (ms)	Ranking
This Work	2800	3200	4600	29000	1st
(Konjaang & Xu, 2021)	4500	4900	5000	32000	2nd

The makespan performance of the proposed approach is presented in Table 1 across four SIGHT workflow categories, with values measured in milliseconds (ms). For Sight Light, the proposed strategy achieved a makespan of 2800 ms compared to 4500 ms for the benchmark, representing a 37.8% reduction in execution time. In the Sight Median scenario, the proposed method recorded 3200 ms versus 4900 ms, yielding a 34.7% improvement, which demonstrates its efficiency under moderate workloads. For Sight Large, the makespan was reduced from 5000 ms to 4600 ms, corresponding to an 8.0% decrease, indicating consistent optimization as workflow size increases. In the Sight Heavy case, the proposed approach achieved 29,000 ms compared to 32,000 ms for the benchmark, resulting in a 9.4% reduction, highlighting its robustness in handling computationally intensive tasks. Overall, the proposed method achieved an average makespan reduction of approximately 22.5% across all scenarios, and consistently ranked first in all workflow categories. These results provide strong quantitative evidence of the method's effectiveness in minimizing makespan and improving workflow scheduling efficiency in cloud computing environments.

The comparison between the benchmark work and the proposed work results for Sight workflow distribution is illustrated in Figure 5 below:

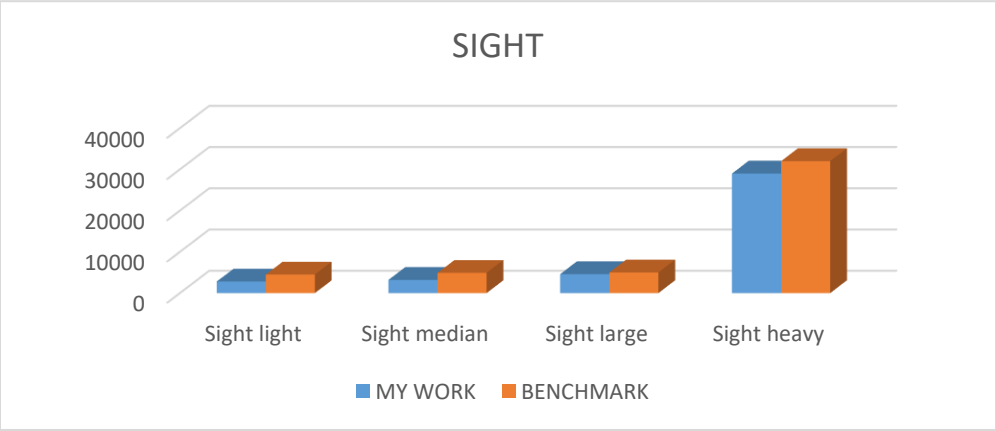

Figure 5: Makespan of SIGHT

iii. Cybershake Workflow

The Cybershake workflow was the second evaluated, chosen because it provides insight into how scheduling strategies manage workloads that grow significantly in complexity and computational requirements. This workflow is critical for testing the adaptability of the proposed scheduling algorithm. The results, as shown in Figure 2 and Figure 2**,** compare the performance of the suggested method under four workload settings in comparison to the benchmark.

Table 2: Makespan of Cybershake

	Cybershake light (ms)	Cybershake light Median (ms)	Cybershake light large (ms)	Cybershake light Heavy (ms)	Ranking
This Work	1500	4800	5500	34000	1st
(Konjaang & Xu, 2021).	4000	6000	7000	40000	2nd

The makespan performance of the proposed approach is presented in Table 2 across four Cybershake workflow categories, with values measured in milliseconds (ms). For Cybershake Light, the proposed strategy achieved a makespan of 1500 ms compared to 4000 ms for the benchmark, representing a 62.5% reduction in execution time. In the Cybershake Light Median scenario, the proposed method recorded 4800 ms versus 6000 ms, yielding a 20.0% improvement, demonstrating its efficiency under moderate workloads. For Cybershake Light Large, the makespan was reduced from 7000 ms to 5500 ms, corresponding to a 21.4% decrease, indicating consistent optimization as workflow size increases. In the Cybershake Light Heavy case, the proposed approach achieved 34,000 ms compared to 40,000 ms for the benchmark, resulting in a 15.0% reduction, highlighting its robustness in handling computationally intensive tasks. Overall, the proposed method achieved an average makespan reduction of approximately 29.7% across all scenarios, and consistently ranked first in all workflow categories. These results provide strong quantitative evidence of the method's effectiveness in minimizing makespan and improving workflow scheduling efficiency in cloud computing environments.

Likewise, Figure 6 presents a comparison between the proposed work result for Cybershake and the benchmark work.

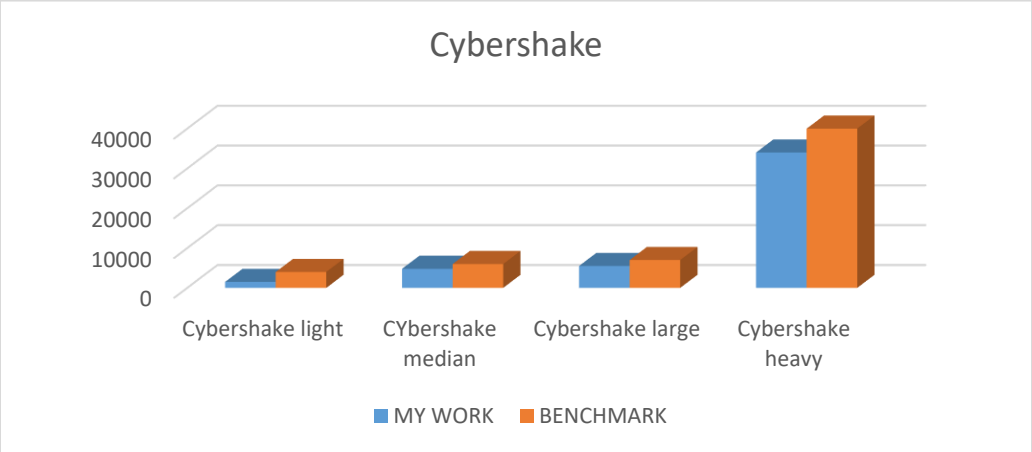

Figure 6: Makespan of Cybershake

iv. Inspiral Workflow

The Inspiral workflow was the final one tested, selected because it provides a strong benchmark for workflows that require high levels of scalability and resource allocation efficiency. This workflow evaluates how well the scheduling algorithm performs in both low and high computational demand environments. The results presented in Figure 7 and Table 3 highlight the differences in makespan between the proposed strategy and the benchmark.

Table 3: Makespan of Inspiral

	Inspiral light (ms)	Inspiral Median (ms)	Inspiral large(ms)	Inspiral Heavy (ms)	Ranking
This Work	2100	2400	2700	38000	1st
(Konjaang & Xu, 2021)	2500	2700	2800	40000	2nd

The makespan performance of the proposed approach is presented in Table 3 across four Inspiral workflow categories, with values measured in milliseconds (ms). For Inspiral Light, the proposed strategy achieved a makespan of 2100 ms compared to 2500 ms for the benchmark, representing a 16.0% reduction in execution time. In the Inspiral Median scenario, the proposed method recorded 2400 ms versus 2700 ms, yielding an 11.1% improvement**,** demonstrating its effectiveness under moderate workloads. For Inspiral Large, the makespan was reduced from 2800 ms to 2700 ms, corresponding to a 3.6% decrease, indicating consistent optimization even as workflow complexity increases. In the Inspiral Heavy case, the proposed approach achieved 38,000 ms compared to 40,000 ms for the benchmark, resulting in a 5.0% reduction, highlighting its robustness in handling computationally intensive tasks. Overall, the proposed method achieved an average makespan reduction of approximately 8.9% across all scenarios, and consistently ranked first in all workflow categories. These results provide strong quantitative evidence of the method's effectiveness in minimizing makespan and improving workflow scheduling efficiency in cloud computing environments.

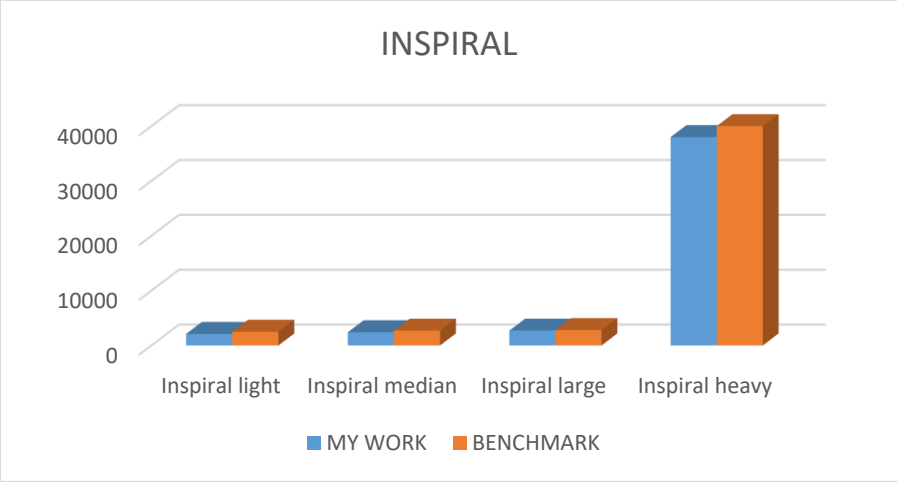

Figure 7: Makespan of Inspiral

RESULT DISCUSSION

This study aimed to minimize makespan and improve resource utilization in cloud environments through an efficient scheduling strategy. The results presented in Tables 1, 2, and 3 demonstrate that the proposed method successfully achieved these objectives by consistently outperforming the benchmark across all evaluated scenarios. In the SIGHT workflow, makespan was reduced from 4500 ms to 2800 ms (37.8% reduction) in the light workload and from 32,000 ms to 29,000 ms (9.4% reduction) in the heavy workload, indicating strong performance across varying task intensities. Similarly, in the Cybershake workflow, makespan decreased significantly from 4000 ms to 1500 ms (62.5% reduction) under light conditions and from 40,000 ms to 34,000 ms (15.0% reduction) under heavy workloads, reflecting substantial efficiency gains and improved resource utilization. For the Inspiral workflow, the proposed algorithm achieved consistent, though comparatively moderate, improvements, reducing makespan from 2500 ms to 2100 ms (16.0% reduction) in the light scenario and from 40,000 ms to 38,000 ms (5.0% reduction) in the heavy scenario. Across all twelve evaluated scenarios, the proposed method achieved an overall average makespan reduction of approximately 20–25**%**, while consistently ranking first in performance. These results provide strong quantitative evidence that the proposed scheduling strategy effectively minimizes execution time, enhances load balancing, and improves resource utilization. Consequently, the study fulfills its research objectives by delivering a robust, scalable, and efficient deadline-aware scheduling approach that demonstrates clear superiority over the benchmark in diverse cloud workflow environments

REFERENCES

1. Abdalkafor, A. S., Jihad, A. A., & Allawi, E. T. (2021). A cloud computing scheduling and its evolutionary approaches. *Indonesian Journal of Electrical Engineering and Computer Science*, *21*(1), 489-496.
2. Ahmad, S. G., Iqbal, T., Munir, E. U., & Ramzan, N. (2023). Cost optimization in cloud environment based on task deadline. *Journal of Cloud Computing*, *12*(1). https://doi.org/10.1186/s13677-022-00370-x
3. Auna Y, S., and Ambursa U, F., (2021) Efficient Max-Min Algorithm for Scheduling Workflow Tasks in Cloud Environment. *International Journal of Information Processing and Communication (IJIPC)* Vol. 10 No. 1&2 [December, 2020], pp. 99-106
4. Beltrán, F., Finardi, E. C., & de Oliveira, W. (2021). Two-stage and multi-stage decompositions for the medium-term hydrothermal scheduling problem: A computational comparison of solution techniques. *International Journal of Electrical Power & Energy Systems*, *127*, 106659.
5. Bothra, S. K., Singhal, S., & Goyal, H. (2021). Deadline-constrained cost-effective load-balanced improved genetic algorithm for workflow scheduling. *International Journal of Information Technology and Web Engineering*, *16*(4), 1–34. https://doi.org/10.4018/IJITWE.2021100101
6. Fan, G., Chen, X., Li, Z., Yu, H., & Zhang, Y. (2023). An Energy-Efficient Dynamic Scheduling Method of Deadline-Constrained Workflows in a Cloud Environment. *IEEE Transactions on Network and Service Management*, *20*(3), 3089–3103. https://doi.org/10.1109/TNSM.2022.3228402

7. Haidri, R. A., Alam, M., Shahid, M., Prakash, S., & Sajid, M. (2022). A deadline aware load balancing strategy for cloud computing. *Concurrency and Computation: Practice and Experience*, *34*(1), 1–16. https://doi.org/10.1002/cpe.6496
8. He, X., Shen, J., Liu, F., Wang, B., Zhong, G., & Jiang, J. (2022). A two-stage scheduling method for deadline-constrained task in cloud computing. *Cluster Computing*, *25*(5), 3265–3281. https://doi.org/10.1007/s10586-022-03561-y
9. Hosseini Shirvani, M., & Noorian Talouki, R. (2022). Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. *Complex & Intelligent Systems*, *8*(2), 1085-1114.
10. Kadhim, A. R., & Rabee, F. (2023). Deadline and Cost Aware Dynamic Task Scheduling in Cloud Computing Based on Stackelberg Game. *International Journal of Intelligent Engineering and Systems*, *16*(3), 175–188. https://doi.org/10.22266/ijies2023.0630.14
11. Khaledian, N., Khamforoosh, K., Akraminejad, R., Abualigah, L., & Javaheri, D. (2024). An energy-efficient and deadline-aware workflow scheduling algorithm in the fog and cloud environment. *Computing*, *106*(1), 109–137. https://doi.org/10.1007/s00607-023-01215-4
12. Khaledian, N., Razzaghzadeh, S., Moazzami, S., & Kivi, P. N. (2025). TM-MOAOA: a two-stage task scheduling approach using TOPSIS and multi-objective Archimedes optimization in fog-cloud environment. In *Computing* (Vol. 107, Issue 7). Springer Vienna. https://doi.org/10.1007/s00607-025-01513-z
13. Konjaang, J. K., & Xu, L. (2021). Multi-objective workflow optimization strategy (MOWOS) for cloud computing. *Journal of Cloud Computing*, *10*(1), 11.
14. Malik, N., Sardaraz, M., Tahir, M., Shah, B., Ali, G., & Moreira, F. (2021). Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds. *Applied Sciences*, *11*(13), 5849.
15. Nabi, S., Aleem, M., Ahmed, M., Islam, M. A., & Iqbal, M. A. (2022). RADL: a resource and deadline-aware dynamic load-balancer for cloud tasks. In *Journal of Supercomputing* (Vol. 78, Issue 12). Springer US. https://doi.org/10.1007/s11227-022-04426-2
16. Raeisi-Varzaneh, M., Dakkak, O., Fazea, Y., & Kaosar, M. G. (2024). Advanced cost-aware Max–Min workflow tasks allocation and scheduling in cloud computing systems. *Cluster Computing*, *27*(9), 13407–13419. https://doi.org/10.1007/s10586-024-04594-1
17. Ramathilagam, A., & Vijayalakshmi, K. (2021). Workflow scheduling in cloud environment using a novel metaheuristic optimization algorithm. *International Journal of Communication Systems*, *34*(5), e4746.
18. Sana, M. U., & Li, Z. (2021). Efficiency aware scheduling techniques in cloud computing: A descriptive literature review. *PeerJ Computer Science*, *7*, 1–37. https://doi.org/10.7717/PEERJ-CS.509
19. Sharma, G., Miglani, N., & Kumar, A. (2021). PLB: a resilient and adaptive task scheduling scheme based on multi-queues for cloud environment. *Cluster Computing*, *24*(3), 2615–2637. https://doi.org/10.1007/s10586-021-03280-w