Journal of Information Systems Engineering and Management
2025, 10(24s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Optimizing Workflow Scheduling in Cloud Computing Through Comparative Study of Dynamic Group and Prioritize Scheduling (DGPS) and Traditional Algorithms

K.Subba Shankar[1],Veeraswamy Ammisetty[2]
[1]*Research Scholar, Department of CSE, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P– 522302.*
[2]*Associate Professor, Department of CSE, Koneru Lakshmaiah Education Foundation, Green Fileds, Vaddeswaram, A.P– 522302.*
E-mail:[1]*kambamshankar@gmail.com,*[2]*ammisetty.veeraswamy@gmail.com*

ARTICLE INFO

Received: 16 Dec 2024

Revised: 02 Feb 2025

Accepted: 20 Feb 2025

ABSTRACT

In cloud computing environments, efficient workflow scheduling is critical for optimizing resource utilization and minimizing response times. This study introduces and evaluates a new scheduling algorithm—Dynamic Group and Prioritize Scheduling (DGPS)—and compares its performance with three traditional algorithms: First-Come-First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR). The DGPS algorithm dynamically groups tasks based on their attributes and prioritizes them before allocation to Virtual Machines (VMs), aiming to enhance scheduling efficiency. Through simulations with 5 VMs and 50 tasks, the performance metrics of average response time and standard deviation were analyzed. The results indicate that DGPS provides a balanced performance with stable response times, while SJF achieves the lowest average response time but with moderate variability. FCFS offers slightly better response times than DGPS but with higher variability, and RR demonstrates the highest response times and standard deviations. This research highlights the effectiveness of DGPS in achieving consistent and efficient task scheduling in cloud environments.

Keywords: Cloud Computing, Dynamic Group and Prioritize Scheduling (DGPS), Performance Optimization, Scheduling Algorithms, Virtual Machines (VMs)

1. INTRODUCTION

Cloud computing has rapidly evolved into a critical component of distributed computing, offering scalable and flexible services via the Internet through hardware and software virtualization. This model enables customers to access services under Service-Level Agreements (SLAs), with a pay-as-you-go pricing system similar to traditional utilities [1]. The key advantages of cloud computing include its adaptability and flexibility, allowing individuals to access and utilize resources tailored to their specific requirements from any remote location [2]. Cloud service providers generally offer two main types of resource provisioning plans to meet various user needs [3]. The first is the on-demand plan, where resources are requested as needed, suitable for unpredictable and fluctuating demand patterns. The second is a reservation-based plan, where users reserve resources in advance, providing a more stable and predictable approach to resource allocation. Industry-leading cloud platforms like Amazon EC2 and GoGrid offer services that support both these plans, accommodating different user requirements [4, 5].

In heterogeneous distributed computing environments, a network of diverse computers, machines, and processors collaborates using high-speed networks to handle complex computational tasks [6]. Previously, scientific applications were executed using grid computing infrastructures, often referred to as e-science [7]. However, the emergence of cloud computing has led to its adoption in e-business and e-science, thanks to its broad availability, cost-effectiveness, and the flexibility offered through virtualization technologies. Cloud computing allows users to execute scientific workflows, which are parallelizable mathematical processes implemented in real-world engineering tasks such as Fast Fourier Transform (FFT), Gaussian-Jordan (GJ) elimination, and LU decomposition. These workflows are typically modeled using Directed Acyclic Graphs (DAGs), where nodes represent different application tasks and edges denote data dependencies among these tasks [8]. Since the shapes and resource demands of these applications can significantly vary, cloud computing elasticity enables it to meet

these diverse needs effectively, providing a scalable solution in cases where users cannot expand their current infrastructure [9].

In cloud environments, multiple virtual machines (VMs) can execute independent tasks simultaneously, enhancing resource utilization. A critical performance metric in this context is the turnaround time, which represents the total elapsed time from the beginning of the first task to the completion of the final one. This metric, commonly referred to as makespan, directly impacts user experience and serves as a primary objective in workflow optimization [10]. Therefore, minimizing makespan is crucial in optimizing resource utilization and enhancing the overall efficiency of cloud-based applications.

Effective execution of scientific workflows in a cloud environment heavily depends on resource allocation strategies, known as workflow scheduling. This process involves distributing workflow tasks to appropriate computing resources while considering various resource priority constraints [11]. Given the NP-complete nature of workflow scheduling, researcher's focus on finding near-optimal solutions rather than absolute ones. To streamline this process, a Workflow Management System (WMS) is essential for defining and managing workflows for execution in cloud environments. The WMS includes a workflow scheduler that acts as a bridge between the workflow tasks and the cloud's computing resources. As shown in Figure 1, this scheduler is responsible for organizing workflow tasks and allocating them to targeted resources in an efficient manner, which is crucial for optimal resource utilization and the successful execution of scientific workflows.

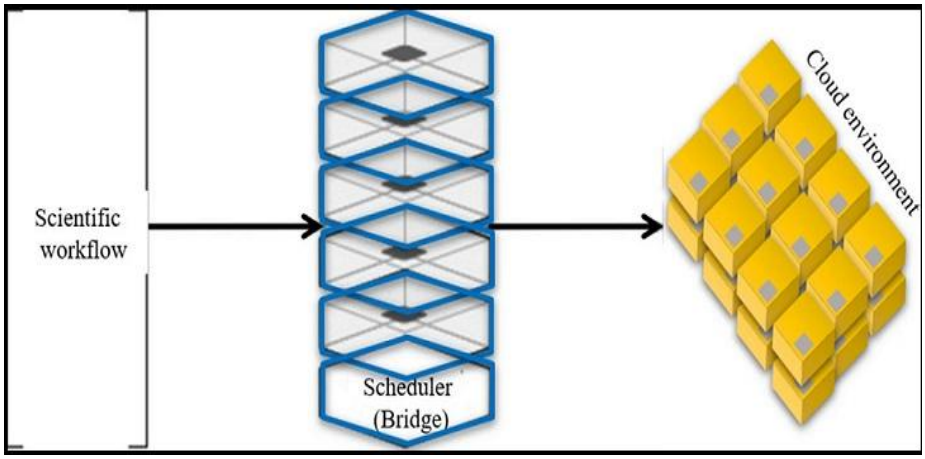

Figure 1: Scientific workflow execution model in cloud computing

Task scheduling in cloud computing is a critical challenge, particularly when dealing with complex and diverse scientific workflows. These workflows often involve large volumes of data, intricate processing requirements, and multiple criteria that must be satisfied simultaneously [12]. The complexity inherent in scientific workflows has led researchers to develop strategies aimed at optimizing their management. One of the primary focuses in these strategies is balancing two often conflicting Quality of Service (QoS) variables: cost and time [13]. QoS serves as a measure of user satisfaction with cloud services and is typically evaluated based on criteria such as reliability, computational cost, and execution time.

The challenge of balancing the dual objectives of minimizing processing time and reducing costs is significant in cloud environments. Faster processing generally necessitates the use of more powerful and, therefore, more expensive resources, whereas opting for cheaper resources might prolong the completion time of tasks [14]. To address this issue, it is essential to devise strategies that not only shorten processing times but also reduce costs, while simultaneously adhering to established deadlines and budgets. The ultimate objective is to find an ideal balance between meeting performance requirements and optimizing resource utilization in a cost-effective manner.

The study in question aims to enhance scientific workflow scheduling performance in cloud computing environments by concentrating on three primary objectives: optimizing execution time, minimizing financial costs, and maintaining effective load balancing across available resources. This study proposes the use of the Whale Optimization Algorithm (WOA) in a multi-level approach to achieve these objectives. The WOA-based approach is designed to minimize the makespan of workflows, ensuring that tasks are completed in the shortest possible time.

By minimizing makespan, users can benefit from improved system responsiveness, directly impacting their overall experience.

In addition to reducing execution time, the proposed method also focuses on minimizing the monetary costs associated with resource utilization. This cost minimization is achieved by optimizing the distribution of tasks among virtual machines (VMs). Effective task distribution not only reduces costs but also improves resource utilization, ensuring that computational resources are used efficiently. Lastly, the study emphasizes the importance of load balancing, which involves distributing the computational load evenly across all available resources. Effective load balancing helps to prevent bottlenecks, thereby enhancing overall system performance and reliability.

To address workflow scheduling issues in dynamic cloud environments, there is a need for an efficient and effective scheduling algorithm. In this study, the proposed Dynamic Group and Prioritize Scheduling (DGPS) algorithm is introduced as a novel solution for optimizing the allocation of tasks in cloud computing. The DGPS algorithm focuses on grouping tasks based on their characteristics, such as priority, resource demands, and deadlines, and then prioritizes these grouped tasks before assigning them to VMs. This structured approach allows for a more efficient allocation of resources, ensuring that critical tasks are executed promptly while less urgent tasks are queued appropriately.

In addition to DGPS, the study also implements and compares several traditional scheduling algorithms, including First-Come-First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR). FCFS is a straightforward scheduling method where tasks are executed in the order they arrive, regardless of their size or resource demands. While simple, FCFS often leads to inefficiencies, especially when larger tasks occupy resources for extended periods, causing delays for smaller, potentially more urgent tasks. On the other hand, SJF prioritizes tasks with the shortest execution time, minimizing the overall processing time but potentially leading to the starvation of longer tasks if shorter ones keep entering the system. The Round Robin (RR) method allocates a fixed time slice for each task, ensuring that all tasks receive an equal opportunity for execution. However, RR can be inefficient for workflows with varying resource demands, as it may not adapt to the specific needs of each task.

The proposed DGPS algorithm aims to address the limitations of these traditional scheduling methods by incorporating a more dynamic and flexible approach. It groups tasks based on their specific attributes and assigns them priorities, ensuring that resource allocation is optimized to meet varying workflow requirements. By doing so, DGPS enhances system performance, minimizes response times, and maximizes resource utilization.

Problem Definition: In cloud computing environments, multiple users simultaneously request services, and these requests are managed by numerous virtual machines (VMs). The primary challenge in this context is to allocate tasks efficiently in order to minimize response time and ensure optimal resource usage. Ineffective task scheduling can lead to resource bottlenecks, increased processing times, and higher operational costs. The proposed DGPS algorithm addresses these issues by grouping tasks according to their characteristics, assigning them appropriate priorities, and then allocating them to VMs in a manner that optimizes both processing speed and cost.

The comparison of DGPS with traditional algorithms such as FCFS, SJF, and RR in this study provides valuable insights into the effectiveness of different scheduling methods in cloud environments. By analyzing the strengths and weaknesses of each algorithm, the study aims to identify the most suitable strategies for various cloud computing scenarios. This approach not only contributes to the understanding of task scheduling in cloud computing but also guides the development of future algorithms that can adapt to the dynamic nature of cloud environments.

In conclusion, task scheduling remains a significant concern in cloud computing, especially when dealing with complex scientific workflows. The proposed DGPS algorithm, combined with a comparative analysis of traditional scheduling methods, offers a comprehensive approach to optimizing workflow execution. By focusing on reducing execution time, minimizing costs, and maintaining load balance, this study aims to enhance the performance and efficiency of cloud-based scientific workflows, ultimately improving user satisfaction and resource utilization.

2. LITERATURE REVIEW

Cloud computing offers a flexible, scalable, and efficient platform for users to access services and resources. Task scheduling plays a pivotal role in optimizing the allocation and utilization of these resources, ultimately improving

the system's performance. Over the years, numerous task scheduling algorithms have been proposed, each addressing various aspects of performance, resource utilization, cost, and energy efficiency. This literature review provides a comprehensive analysis of these task scheduling algorithms and their methodologies.

The "Cost-Based Task Scheduling Algorithm" proposed by Garg [15] focuses on efficiently allocating tasks in a cloud computing environment, particularly at the platform and infrastructure levels. Cloud computing environments often face the challenge of assigning 'm' tasks to 'n' virtual machines (VMs) where 'm' exceeds 'n'. The algorithm aims to optimize resource usage by evaluating the processing cost of each task on every VM using the Shortest Job First (SJF) approach. The SJF algorithm allocates tasks to VMs based on the minimum processing load to ensure optimal resource utilization. The process begins by reading the number of VMs (n) and tasks (m), then calculates the processing cost for each task on the available VMs. Tasks are then assigned to the VM with the minimum processing load iteratively until all tasks are scheduled. This method helps in reducing total processing costs and optimizing task execution times. However, the primary limitation of this approach is that it assumes a known processing cost for each task, making it less adaptable in dynamic and unpredictable cloud environments.

Nandhini, Radha, Pavithra, and Srikanth [17] presented a comprehensive "Survey on Task Scheduling Models Using Optimization Techniques" to address task scheduling challenges in cloud computing. The study highlights that the primary goal of a task scheduling algorithm is to minimize makespan (the total time required to complete a set of tasks) and maximize resource utilization. Various algorithms, including Max-Min, Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Ant Colony Algorithm, and Bee Colony Algorithm, were discussed in their paper. The survey introduces a Hybrid Cuckoo Algorithm that combines the advantages of the Genetic Algorithm and Cuckoo Algorithm, aiming to improve energy efficiency, execution time, and resource utilization. This hybrid approach effectively eliminates the need for traditional task scheduling algorithms, thereby reducing overall scheduling time. Despite its advantages, the complexity of implementing hybrid algorithms and their computational costs are noteworthy challenges that need to be considered in real-world cloud scenarios.

The "Enhanced Max-Min Task Scheduling Algorithm" proposed by Bhoi and Ramanuj [20] emphasizes the need for reduced waiting times, reduced makespan, optimal resource utilization, and overall better system performance. The proposed Scheduling Algorithm (SA) aims to enhance traditional scheduling approaches by improving the task allocation process. Using the CloudSim framework for evaluation, the SA algorithm demonstrated superior performance in comparison to the existing SJF algorithms. By consistently reducing processing times, the algorithm showcased its potential in optimizing task scheduling within cloud environments. However, the limitations of this approach include the lack of scalability in large-scale cloud systems and the assumption of homogeneous resources.

Saxena, Chauhan, and Kait [21] introduced the "Dynamic Fair Priority Optimization Task Scheduling Algorithm," which implements the concept of "Weighted Fair Queuing" to enhance quality of service (QoS). This algorithm addresses challenges related to resource allocation, task execution order, overhead minimization, VM monitoring, and cost considerations in cloud task scheduling. The algorithm categorizes tasks into two groups: deadline-based and reduced cost-based, using dynamic optimization and priority equity principles. It utilizes three priority queues (high, mid, low) with assigned weights, implementing a round-robin approach. This algorithm's key benefit is its ability to balance fairness and efficiency for both users and service providers. However, it introduces complexity due to the classification and dynamic priority adjustments, which may lead to increased computational overhead.

The "Task Scheduling Algorithm with Improved Makespan Based on Prediction of Task Computation Time (PTCT)" proposed by Al-Maytami, Fan, Hussain, Baker, and Liatsis [23] introduces a novel approach using Directed Acyclic Graphs (DAGs) and Principal Component Analysis (PCA). The PTCT algorithm focuses on enhancing task scheduling performance and minimizing computational costs in cloud environments. By leveraging PCA to minimize matrix size, PTCT efficiently optimizes resource allocation and considers QoS constraints. It was compared to other state-of-the-art scheduling algorithms like Min-Min, Max-Min, QoS-Guide, and Min-Max, demonstrating superior performance in terms of speedup, efficiency, and schedule length ratio. Despite its advantages, PTCT's fixed approach may limit flexibility in highly dynamic cloud environments, and its reliance on historical data for prediction may not always guarantee optimality.

The "Dynamic Priority Scheduling Algorithm Based on Heapsort," proposed by Meng, Zhu, and Xia [25], introduces a new method that accounts for task deadlines, values, and energy consumption. Using the hierarchy process

(FAHP) to prioritize tasks, the algorithm employs heapsort to efficiently sort tasks in order of priority. This algorithm is particularly suited for real-time systems and industrial control applications, where reducing the frequency of missed deadlines is crucial. The experimental results indicate a decrease in missed deadlines by an average of 0.1789, thereby enhancing overall scheduling performance. However, this algorithm's reliance on heapsort can become a bottleneck in environments with rapidly changing task priorities, impacting real-time performance.

2.1 Comparison of Methodologies and Drawbacks

To provide a more structured overview of these methodologies, a comparison table (Table 1) is provided below. This table outlines each algorithm, its methodology, and its key drawbacks/limitations.

Table 1: Comparison of Related Work and Methodologies

Author	Paper Title	Methodology	Drawbacks/Limitations
Meng, S., et.al [25]	Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort	Heapsort, Dynamic Priority Scheduling	Bottleneck in real-time environments, High computational complexity
Shi, J., et.al. [26]	Elastic Resource Provisioning for Scientific Workflow Scheduling in Cloud Under Budget and Deadline Constraints	Budget-Constrained Scheduling, Elastic Resource Provisioning	Limited to budget and deadline constraints, Less adaptable to real-time changes.
Aziza, et.al [27]	A Hybrid Genetic Algorithm for Scientific Workflow Scheduling in Cloud Environment	Hybrid Genetic Algorithm	High computational overhead, Requires fine-tuning of parameters
Iranmanesh, et.al [28]	DCHG-TS: A Deadline-Constrained and Cost-Effective Hybrid Genetic Algorithm for Scientific Workflow Scheduling in Cloud Computing.	Hybrid Genetic Algorithm, Deadline-Constrained Scheduling.	High complexity, Scalability concerns.
Mohammadzadeh, A., et.al [29]	Scientific Workflow Scheduling in Multi-Cloud Computing Using a Hybrid Multi-Objective Optimization Algorithm.	Hybrid Multi-Objective Optimization	Limited real-world validation, High computational cost.
Choudhary, et.al [30]	Energy-Aware Scientific Workflow Scheduling in Cloud Environment.	Energy-Aware Scheduling, Workflow Optimization.	Complex implementation, Limited flexibility in dynamic environments.
Khaleel, et.al [31]	Multi-Objective Optimization for Scientific Workflow Scheduling Based on Performance-to-Power Ratio in Fog–Cloud Environments.	Multi-Objective Optimization, Performance-to-Power Ratio.	Limited applicability to large-scale cloud environments, Requires accurate power-performance models.
Al-Moalmi, et.al. [32]	A Whale Optimization System for Energy-Efficient Container Placement in Data Centers	Whale Optimization Algorithm	High computational complexity, Limited evaluation in diverse scenarios

In summary, task scheduling in cloud computing has been extensively explored using various algorithms and techniques. Each approach brings unique advantages while facing its own set of challenges. The evolution from traditional cost-based scheduling to more dynamic and optimized algorithms signifies the continuous effort to improve cloud resource management. The next step in this research involves comparing the proposed Dynamic Group and Prioritize Scheduling (DGPS) approach with these traditional methodologies to identify potential enhancements in performance, resource utilization, and overall efficiency.

3. PROPOSED METHODOLOGY

The proposed methodology aims to address workflow scheduling issues in cloud computing environments with dynamic resource availability. Given the cloud's vast network of servers and the continuous influx of user tasks, scheduling needs to be optimized to ensure that each task receives adequate resources promptly. The methodology focuses on dynamically grouping tasks based on their characteristics, prioritizing them, and allocating the most

suitable Virtual Machine (VM) for execution. This systematic approach enhances resource utilization, minimizes response time, and efficiently manages the execution of diverse tasks.

3.1. Task Grouping Based on Attributes

The first step of the proposed methodology involves grouping incoming tasks based on specific attributes such as deadlines, cost requirements, or resource demands. This dynamic grouping reduces the complexity of scheduling by categorizing tasks with similar characteristics, allowing for more targeted resource management. The attribute-based clustering helps in the effective distribution of tasks, optimizing their placement and resource usage across the cloud environment. The key aspects of this phase include:

a) **Dynamic Identification of Attributes:** Each task is analyzed to identify attributes like execution time, cost sensitivity, and urgency (e.g., deadline constraints).

b) **Attribute-Based Clustering:** Tasks are then dynamically grouped based on these attributes. For example, tasks with strict deadlines are grouped separately from those that are cost-sensitive, while tasks requiring high computational resources are clustered based on resource demand.

3.2. Task Prioritization within Groups

After grouping, the methodology involves prioritizing tasks within each group. This prioritization ensures that the most critical tasks receive attention first, thereby optimizing response times and resource allocation. The prioritization is based on a scoring mechanism that incorporates various task attributes:

a) Priority Score Calculation:

A priority score is assigned to each task using a weighted combination of its attributes:

$$\textbf{Priority Score} = w1 \times \textbf{Deadline Urgency} + w2 \times \textbf{Execution Time} + w3 \times \textbf{Cost Sensitivity} \quad (1)$$

Where:

- w1, w2, and w3 are adjustable weights based on system requirements
- Deadline Urgency represents deadline proximity
- Execution Time represents task duration
- Cost Sensitivity represents resource usage

b) Sorting by Priority:

Tasks within each group are sorted in descending order of their priority scores, ensuring:

a) Most critical tasks are addressed promptly

b) Efficient management of other tasks

By prioritizing tasks based on their attributes, this methodology optimizes resource allocation and response times.

3.3. Dynamic Resource Allocation to Virtual Machines

The final step involves dynamically allocating tasks to the available VMs based on their capabilities and current load status. The cloud environment typically consists of multiple VMs, each with varying resource capacities. The methodology selects the optimal VM for each task, considering both task priority and VM status:

a) **VM Selection:** The system continuously monitors the available VMs, evaluating their response times and resource capacities. For each task, the VMs capable of handling it are identified, and the one with the minimum response time is selected for execution.

b) **Task Assignment and Execution:** The selected VM is then assigned the task, and its status is updated to reflect the new workload. This dynamic assignment helps maintain an optimal balance across the cloud infrastructure, preventing overloading of individual VMs and improving overall system performance.

c) **Feedback Mechanism:** The methodology includes a feedback mechanism that continuously monitors VM performance. If any inefficiencies or delays are detected, adjustments are made to future task assignments, ensuring ongoing optimization of resource usage and response times.

3.3 Proposed Algorithm: Dynamic Group-Prioritized Scheduling (DGPS) Algorithm

To design a dynamic task scheduling algorithm that optimizes workflow management by grouping tasks based on specific attributes, prioritizing them, and allocating dynamic resources (VMs) in a cloud environment to minimize response time and maximize resource utilization.

Steps of the Algorithm:

---

1. Input:

- Set of incoming tasks: $T = \{T1, T2, ..., Tn\}$

- Set of available Virtual Machines (VMs): $V = \{V1, V2, ..., Vm\}$

- Task attributes: deadlines, cost requirements, resource demands

- VM attributes: response time, current load, resource capacity

2. Initialize:

- Create empty groups $G = \{G1, G2, ..., Gk\}$ based on task attributes (e.g., deadlines, cost requirements)

- Initialize a priority queue for each group

3. Task Grouping:

- For each incoming task Ti in T:

- Identify its attributes (e.g., deadline, cost)

- Assign Ti to the appropriate group Gj based on its attributes

4. Task Prioritization:

- For each group Gj in G:

- Calculate a priority score for each task Ti in Gj using the formula:

Priority Score = $w1 \times$ Deadline Urgency + $w2 \times$ Execution Time + $w3 \times$ Cost Sensitivity

- Sort tasks in Gj based on their priority scores in descending order

5. VM Selection and Task Assignment:

- For each task Ti in the priority queue of group Gj:

- Identify the set of VMs $V' \subseteq V$ that have the capacity to execute Ti

- Select the VM $Vmin \in V'$ with the minimum response time

- Assign Ti to Vmin for execution

- Update the status and load of Vmin to reflect the addition of Ti

6. Feedback and Adjustment:

- Continuously monitor the performance of each VM

- Adjust task groupings, priorities, and VM selections based on real-time feedback to optimize future scheduling

7. Output:

- An optimal schedule of tasks assigned to VMs

8. End of Algorithm

This proposed methodology ensures an adaptive and efficient approach to workflow scheduling in cloud environments. By dynamically grouping tasks, prioritizing them, and allocating resources based on real-time system status, it addresses the complexities of handling dynamic resources in cloud computing. The feedback mechanism further enables ongoing optimization, improving system performance and user experience.

4. Implementation

In our study, we implemented and tested four scheduling algorithms to evaluate their performance in a simulated cloud computing environment. The experimental setup included:

- **Number of Virtual Machines (VMs): 5**
- **Number of Tasks: 50**

The tasks were characterized by varying execution times and deadlines to create a realistic workload. The four algorithms under evaluation were:

1. Dynamic Group and Prioritize Scheduling (DGPS)
2. First-Come-First-Served (FCFS)
3. Shortest Job First (SJF)
4. Round Robin (RR)

Each algorithm was implemented and applied to the same set of tasks across the 5 VMs. The tasks were scheduled according to the specific rules of each algorithm, and their performance was measured based on the average response time and standard deviation of response times.

4.1 Algorithm Implementation Details:

a) **DGPS:** Tasks were first grouped based on deadlines and then prioritized within each group by execution time. Each task was assigned to the VM with the minimum load, aiming for efficient resource utilization and timely task completion.

b) **FCFS:** Tasks were processed in the order they arrived. Each task was assigned to the VM with the least current load, without considering task priorities or deadlines.

c) **SJF:** Tasks were prioritized based on their execution times, with shorter tasks processed before longer ones. This method aimed to minimize the average response time by handling shorter tasks first.

d) **RR:** Tasks were allocated to VMs in a cyclic manner with fixed time slices. This ensured each VM received an equal share of CPU time, though it might not be optimal for tasks with varying lengths.

5. RESULTS

The performance of each scheduling algorithm was assessed by measuring the average response time and standard deviation of response times. The results are summarized in the table below:

Algorithm	Average Response Time (units)	Standard Deviation (units)
DGPS	57.61	4.13
FCFS	57.06	5.88
SJF	42.26	4.78
Round Robin (RR)	144.37	22.08

5.1 Performance Analysis:

a) **DGPS:** This algorithm provided a balanced performance with an average response time of 57.61 units and a low standard deviation of 4.13 units. The low standard deviation indicates consistent task handling and stable performance across various scenarios.

b) **FCFS:** The FCFS algorithm achieved a marginally better average response time of 57.06 units compared to DGPS. However, it had a higher standard deviation of 5.88 units, reflecting greater variability in response times and less consistency in scheduling.

c) **SJF:** The SJF algorithm excelled in minimizing the average response time, achieving the lowest value of 42.26 units. Despite its superior average response time, SJF had a moderate standard deviation of 4.78 units, indicating some variability in task completion times.

d) **RR:** The Round Robin algorithm resulted in the highest average response time of 144.37 units and the highest standard deviation of 22.08 units. This indicates that RR was the least efficient, with significant delays and variability in task completion.

To visualize the performance differences among the algorithms, the following graphs illustrate the average response times and standard deviations.

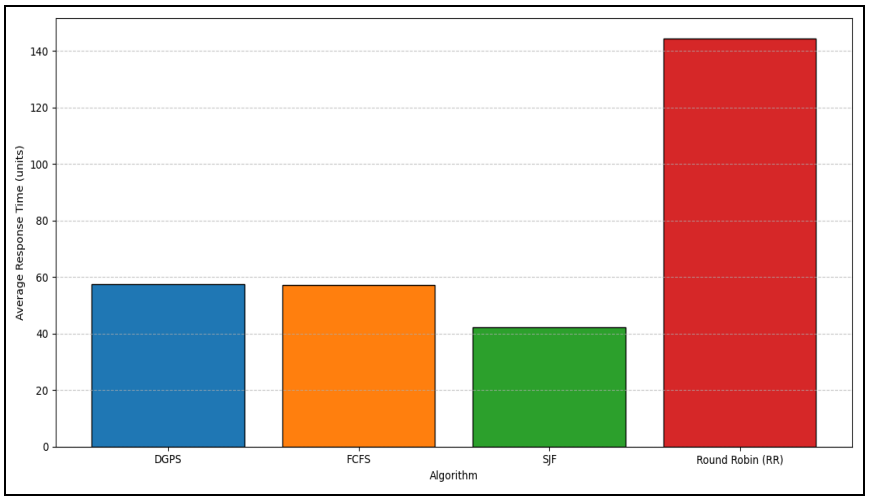

Figure 2: Average Response Time Comparison

The graph shows that SJF has the lowest average response time, followed by DGPS and FCFS. RR has the highest average response time.

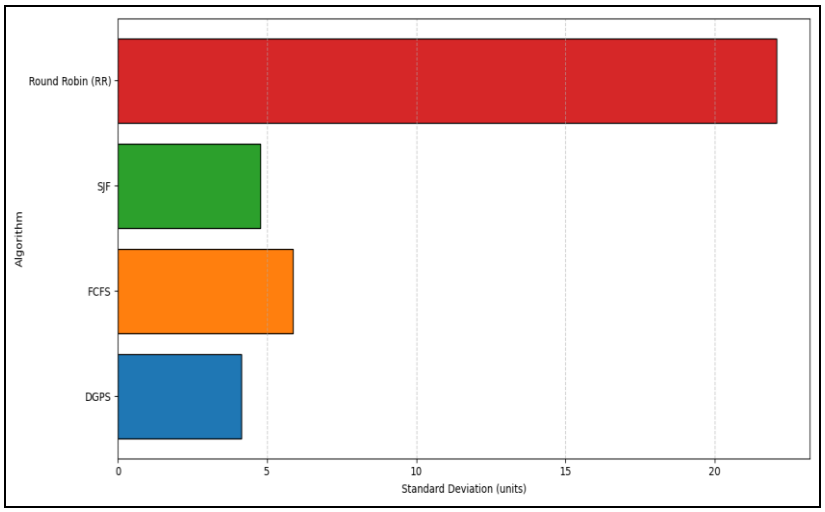

Figure 3: Standard Deviation Comparison

This graph highlights the consistency of the algorithms. DGPS and SJF have lower standard deviations compared to FCFS and RR, indicating more stable performance.

The comparative analysis reveals that the DGPS algorithm offers a good balance between response time and consistency, making it suitable for environments where stable performance is crucial. The SJF algorithm, while providing the best average response time, introduces some variability in response times. The FCFS algorithm, although slightly better than DGPS in response time, suffers from higher variability. The RR algorithm, with its high response times and standard deviation, proves to be less effective in optimizing scheduling performance.

6. CONCLUSION AND FUTURE SCOPE

Efficient workflow scheduling is crucial in cloud computing for optimizing resource use and minimizing response times. This study introduced the Dynamic Group and Prioritize Scheduling (DGPS) algorithm and compared its performance with traditional methods like First-Come-First-Served (FCFS), Shortest Job First (SJF), and Round Robin (RR). DGPS proved effective by providing balanced and stable response times, with SJF showing the lowest average response time but higher variability, and FCFS performing slightly better than DGPS in response time but with increased inconsistency. Round Robin resulted in the highest response times and variability. Future work could enhance DGPS by integrating machine learning for dynamic task management, testing in larger-scale or heterogeneous cloud environments, and incorporating energy-efficient mechanisms to address data center energy consumption.

Funding: No- funding.

Availability of data and materials: Data will be made available on reasonable request.

Code availability: Available on reasonable request.

Declarations Conflict of interest: There is no conflict of interest.

REFERENCES

[1] Hayyolalam, V., Pourghebleh, B., Kazem, A. A. P., & Ghaffari, A. (2019). Exploring the state-of-the-art service composition approaches in cloud manufacturing systems to enhance upcoming techniques. *International Journal of Advanced Manufacturing Technology, 105*(1–4), 471–498.

[2] Wang, X., Sun, Y., Sun, Q., Lin, W., Wang, J. Z., & Li, W. (2023). HCIndex: a Hilbert-curve-based clustering index for efficient multi-dimensional queries for cloud storage systems. *Cluster Computing, 26*(3), 2011–2025.

[3] Hayyolalam, V., Pourghebleh, B., & Chehrehzad, M. R., Pourhaji Kazem, A. A. (2022). Single-objective service composition methods in cloud manufacturing systems: recent techniques, classification, and future trends. *Concurrency and Computation: Practice and Experience, 34*(5), e6698.

[4] Yakubu, I. Z., & Murali, M. (2023). An efficient meta-heuristic resource allocation with load balancing in IoT-Fog-cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing, 14*(3), 2981–2992.

[5] Sefati, S., Mousavinasab, M., & Zareh Farkhady, R. (2022). Load balancing in cloud computing environment using the grey wolf optimization algorithm based on the reliability: performance evaluation. *Journal of Supercomputing, 78*(1), 18–42.

[6] Al-Jumaili, A. H. A., Muniyandi, R. C., Hasan, M. K., Paw, J. K. S., & Singh, M. J. (2023). Big data analytics using cloud computing-based frameworks for power management systems: status, constraints, and future recommendations. *Sensors, 23*(6), 2952.

[7] He, J. (2022). Cloud computing load balancing mechanism taking into account load balancing ant colony optimization algorithm. *Computational Intelligence and Neuroscience, 2022*, 3120883.

[8] Mangalampalli, S., et al. (2023). Prioritized task-scheduling algorithm in cloud computing using cat swarm optimization. *Sensors, 23*(13), 6155.

[9] Praveenchandar, J., & Tamilarasi, A. (2021). Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. *Journal of Ambient Intelligence and Humanized Computing, 12*(3), 4147–4159.

[10] Dubey, K., & Sharma, S. C. (2021). A novel multi-objective CR-PSO task scheduling algorithm with deadline constraint in cloud computing. *Sustainable Computing, 32*, 100605.

[11] Hosseinzadeh, M., Ghafour, M. Y., Hama, H. K., Vo, B., & Khoshnevis, A. (2020). Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. *Journal of Grid Computing, 18*, 1–30.

[12] Kamanga, C. T., Bugingo, E., Badibanga, S. N., & Mukendi, E. M. (2023). A multi-criteria decision-making heuristic for workflow scheduling in cloud computing environment. *Journal of Supercomputing, 79*(1), 243–264.

[13] Mikram, H., El Kafhali, S., & Saadi, Y. (2024). HEPGA: a new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. *Simulation Modelling Practice and Theory, 130*, 102864.

[14] Asghari Alaie, Y., Hosseini Shirvani, M., & Rahmani, A. M. (2023). A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach. *Journal of Supercomputing, 79*(2), 1451–1503.

[15] Garg, S. (2014). Cost-Based Task Scheduling Algorithm in Cloud Computing. *International Journal of Research in Engineering and Technology, 03,* 59-61. https://doi.org/10.15623/ijret.2014.0326013

[16] Chawla, Y., & Bhonsle, M. (2013). Dynamically Optimized Cost-Based Task Scheduling in Cloud Computing. *International Journal of Emerging Trend & Technology in Computer Science (IJETTCS), 2*(3), 38-42.

[17] Nandhini, A., Radha, S., Pavithra, T. V., & Srikanth, G. U. (2017). A Survey on Task Scheduling Model in Cloud Computing Using Optimization Techniques. *International Journal of Advanced Research, 5*(2), 345-348.

[18] Ramya, G., Keerthika, P., Suresh, P., & Sivaranjani, M. (2016). Optimized Scheduling of Tasks Using Heuristic Approach With Cost-Efficiency in Cloud Data Centers. *International Journal of Scientific & Engineering Research, 7*(2), 208-213.

[19] Arora, S., & Anand, S. (2014). Improved Task Scheduling Algorithm in Cloud Environment. *International Journal of Computer Applications, 96*(7), 7-12. https://doi.org/10.5120/16772-6342

[20] Bhoi, U., & Ramanuj, P. N. (2013). Enhanced Max-Min Task Scheduling Algorithm in Cloud Computing. *International Journal of Application or Innovation in Engineering & Management, 2*(4), 259-264.

[21] Saxena, D., Chauhan, R. K., & Kait, R. (2016). Dynamic Fair Priority Optimization Task Scheduling Algorithm in Cloud Computing: Concepts and Implementations. *International Journal of Computer Network and Information Security (IJCNIS), 8*(2), 41-48. https://doi.org/10.5815/ijcnis.2016.02.05

[22] Chaudhary, M., & Peddoju, S. K. (2012). A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment. *International Journal of Engineering Research and Application, 2*(3), 1-5.

[23] Ch, R., Batra, I., & Malik, A. (2023). Blockchain-based secure with improvised bloom filter over a decentralized access control network on a cloud platform. *Journal of Engineering Science and Technology Review, 16*(2), 123–130.

[24] Ravikumar, C. H., Batra, I., & Malik, A. (2022). A novel design to minimise the energy consumption and node traversing in blockchain over cloud using ensemble cuckoo model. *International Journal on Recent and Innovation Trends in Computing and Communication, 10*(1), 254–264.

[25] Al-Maytami, B. A., Fan, P., Hussain, A., Baker, T., & Liatsis, P. (2019). A Task Scheduling Algorithm with Improved Makespan Based on Prediction of Task Computation Time for Cloud Computing. *IEEE Access, 7,* 160916-160926. https://doi.org/10.1109/ACCESS.2019.2948704

[26] Maipan-uku, J. Y., Muhammed, A., Abdullah, A., & Hussin, M. (2016). Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environment. *Journal of Telecommunication, Electronic and Computer Engineering, 8*(6), 43-47.

[27] Meng, S., Zhu, Q., & Xia, F. (2019). Improvement of the Dynamic Priority Scheduling Algorithm Based on a Heapsort. *IEEE Access, 7,* 68503-68510. https://doi.org/10.1109/ACCESS.2019.2917043

[28] Shi, J., Luo, J., Dong, F., Zhang, J., & Zhang, J. (2016). Elastic resource provisioning for scientific workflow scheduling in cloud under budget and deadline constraints. *Cluster Computing, 19*, 167–182.

[29] Aziza, H., & Krichen, S. (2020). A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. *Neural Computing and Applications, 32*, 15263–15278.

[30] Iranmanesh, A., & Naji, H. R. (2021). DCHG-TS: a deadline-constrained and cost-effective hybrid genetic algorithm for scientific workflow scheduling in cloud computing. *Cluster Computing, 24*, 667–681.

[31] Mohammadzadeh, A., & Masdari, M. (2021). Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm. *Journal of Ambient Intelligence and Humanized Computing, 14*, 3509–3529.

[32] Choudhary, A., Govil, M. C., Singh, G., Awasthi, L. K., & Pilli, E. S. (2022). Energy-aware scientific workflow scheduling in cloud environment. *Cluster Computing, 25*(6), 3845–3874.

[33] Ravikumar, C., Naresh, B., Prasanna, P. L., Goud, E. A., & Prasad, P. R. (2024). Exploring machine learning algorithms for robust cyber threat detection and classification: A comprehensive evaluation. In *Proceedings of the 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS 2024)*

[34] Khaleel, M. I. (2022). Multi-objective optimization for scientific workflow scheduling based on performance-to-power ratio in fog–cloud environments. *Simulation Modelling Practice and Theory, 119*, 102589.

[35] Al-Moalmi, A., Luo, J., Salah, A., Li, K., & Yin, L. (2021). A whale optimization system for energy-efficient container placement in data centers. *Expert Systems with Applications, 164*, 113719.