JCTA Journal of Computing Theories and Applications ISSN:3024-9104

*Research Article*

# Enhancing Cloud Task Scheduling with Multi-Objective Optimization Using K-Means Clustering and Dynamic Resource Allocation

**July Lwin ***

Faculty of Computer Science, University of Computer Studies, Hpa-An, Myanmar;
email: julylwin178@gmail.com
* Corresponding Author : July Lwin

**Abstract:** The scheduling and resource allocation procedure is an essential component of cloud resource management. Effective resource allocation is severely hampered by the task arrival rates' erratic and unclear behavior. To prevent under or overusing resources, an effective scheduling strategy is necessary. To improve scheduling and allocation performance, a multi-objective optimization technique is presented for the best resource allocation and task scheduling inside scientific workflow datasets in a heterogeneous environment. In the first stage, the system calculates four key metrics: Communication Cost, Computation Cost, Earliest Finished Time on a particular VM, and Total Task Length for a specific scientific workflow dataset. These metrics provide a comprehensive understanding of the resource requirements and help make informed scheduling decisions. In the second stage, tasks are clustered using the K-Means clustering algorithm. This clustering groups similar tasks together, making managing and scheduling them easier. In the third stage, the proposed resource allocation algorithm allocates the clustered tasks to the appropriate VMs. This step ensures that the tasks are assigned to the best-suited resources, optimizing the overall system performance and resource utilization. By following this multi-stage process, the system aims to achieve optimal resource allocation and task scheduling, thereby improving the efficiency and effectiveness of cloud resource management. The proposed method significantly outperforms PSO, CSO, and GWO by consistently achieving lower Makespan—under 400 units at 50 tasks—while maintaining high resource utilization rates above 0.75, demonstrating superior efficiency in task execution and resource management.

**Keywords:** Cloud resource management; K-Means clustering; Resource allocation; Task scheduling; Resource utilization.

**Received: August, 5th 2024**
**Revised: September, 17th 2024**
**Accepted: September, 22nd 2024**
**Published: October, 1st 2024**



## 1. Introduction

Cloud computing has revolutionized the execution of complex scientific workflows by providing a flexible, scalable, and cost-effective platform for managing large-scale computational tasks. These workflows typically consist of numerous interdependent tasks that require efficient scheduling and resource allocation to optimize performance. However, the inherent unpredictability and variability of task arrival rates in cloud environments pose significant challenges to effective resource management. Inefficient scheduling can lead to underutilization or overutilization of resources, negatively impacting overall system performance. This research suggests a multi-objective optimization technique for heterogeneous cloud environments that optimizes resource allocation and task scheduling in scientific workflow datasets [1], [2]. The proposed method aims to enhance the efficiency of scheduling and resource allocation through a structured, three-stage approach. In the initial stage, the system computes four fundamental metrics for each scientific workflow dataset: Communication Cost, Computation Cost, Earliest Finished Time on a particular Virtual Machine (VM), and Total Task Length. These metrics provide a comprehensive understanding of the resource requirements and execution characteristics of each task, forming the basis for informed scheduling decisions. The second stage involves clustering tasks using the K-Means clustering algorithm. By

grouping similar tasks based on the calculated metrics, this clustering approach facilitates more efficient management and scheduling. Clustering helps to categorize tasks into manageable subsets, improving the overall scheduling process. The suggested resource allocation algorithm is used in the last phase to allocate the clustered jobs to the best-suited virtual machines. This ensures that tasks match the best-suited resources, optimizing performance and resource utilization.

K-means clustering was chosen to enhance cloud task scheduling because it efficiently groups tasks with similar resource requirements, improving the dynamic allocation of resources in large-scale cloud environments. Its simplicity, scalability, and quick convergence make it suitable for handling large datasets commonly encountered in cloud systems. K-means help optimize resource utilization, reduce task completion time, and improve load balancing by clustering tasks based on their resource needs. Combined with multi-objective optimization, this clustering approach enables dynamic resource allocation, where resources can be scaled and distributed in real-time to meet varying workload demands and improve cloud performance efficiency. This paper makes several key contributions to the field of cloud resource management:

- Introduces a novel multi-objective optimization method that balances various conflicting objectives, such as minimizing communication and computation costs while maximizing resource utilization and task completion time.
- Develops a comprehensive metric-based approach to understand and characterize the resource requirements of tasks, providing a solid foundation for effective scheduling.
- Utilizes the K-Means clustering algorithm to group tasks with similar characteristics, enhancing the manageability and efficiency of the scheduling process.
- Proposes a resource allocation algorithm that dynamically assigns tasks to the most suitable VMs, ensuring optimal use of available resources.
- Demonstrates significant improvements in task scheduling and resource allocation performance through extensive experimentation with scientific workflow datasets.

By addressing the challenges of task scheduling and resource allocation in cloud environments, this research contributes to advancing efficient and effective cloud computing solutions, particularly for scientific workflows. The proposed multi-objective optimization method not only enhances performance but also provides a scalable and reliable approach to managing complex computational tasks in heterogeneous cloud environments.

The remaining sections of this document are organized as follows: We explore relevant literature in Section 2 and background theory relevant to the topic in Section 3. An explanation of the suggested system architecture and its constituent parts is provided in Section 4. Subsequently, Section 5 outlines the performance evaluation approach, providing insight into the metrics applied and the experimental design. Section 6 provides a summary of the study's main findings as well as closing thoughts on the overall research.

## 2. Literature Reviews

Cloud computing, characterized by its pay-as-you-use billing model, offers distributed infrastructure and services [3], [4]. The increasing complexity and diversity of scientific workflows present significant issues for cloud providers regarding load balancing and resource provisioning. Despite a wealth of research on workflow scheduling in cloud systems, many techniques fail to consider the necessity of addressing several competing goals. In order to address these problems, a novel workflow scheduling approach utilizing the Firefly Algorithm (FA) was presented in research [5]. This approach considers several competing goals, including Makespan, dependability, server workload, and resource usage. The FA seeks to balance loads and maximize resource utilization by determining which cloud servers suit each process.

Furthermore, a rule-based methodology allocates assignments to appropriate virtual machine instances, reducing makespan and meeting deadlines. Google cluster traces are used to assess the suggested approach, and in-depth simulation runs are conducted to look at the FA's control settings to improve performance. According to experimental results, the suggested approach performs better than current algorithms regarding Makespan, dependability, resource usage, and server load balancing, among other Quality-of-Service (QoS) metrics. Research [6] presented a novel multi-objective minimum weight method for Pareto front computation, enhancing the FR-MOS scheduling algorithm with PSO and fuzzy resource management to optimize diversity. Key objectives include reliability, cost, resource utilization, risk

probability, and Makespan. The proposed minimum weight optimization (MWO) strategy assists in selecting optimal trade-offs among these goals. Evaluated against five decision-making methods using scientific workflows, MWO effectively balances consumer and provider interests, outperforming other strategies regarding competing objectives. To determine the Pareto front, a novel multi-objective minimal weight technique is presented in a study [7] that considers competing goals, including Makespan, cost, dependability, resource consumption, and risk likelihood. The suggested minimal weight optimization (MWO) technique balances the interests of service providers and customers by adaptively calculating inertia weights. This helps identify the best trade-offs. The multi-objective scheduling algorithm with MWO (MOS-MWO) is compared with the conventional multi-objective scheduling method (MOS) using typical scientific workflows. The efficiency of MOS-MWO is demonstrated by the results, which show that it achieves a better QoS satisfaction rate. In order to effectively schedule scientific activities, research [8] presented a unique multi-objective scheduling technique called FR-MOS. It uses particle swarm optimization (PSO) and fuzzy resource management. The algorithm seeks to guarantee reliability while minimizing costs and Makespan. Its coding technique takes into account both the order of data transmission and the task execution location at the same time. Based on performance criteria, simulation findings show that FR-MOS performs much better than the basic MOS algorithm based on PSO. This paper presented a Task Scheduling-Decision Tree (TS-DT), a multi-objective task scheduling technique tailored for various scenarios [9]. A novel method for allocating and executing application tasks more efficiently is introduced: TS-DT. In performance testing, TS-DT was compared with three different algorithms: Heterogeneous Earliest Finish Time (HEFT), Q-Learning in conjunction with HEFT (QL-HEFT), and Technique for Order of Preference by Similarity to Ideal Solution with the Entropy Weight Method (TOP-SIS-EWM). The results demonstrate that TS-DT outperforms HEFT, TOPSIS-EWM, and QL-HEFT by reducing Makespan by 5.21%, 2.54%, and 3.32%, increasing resource usage by 4.69%, 6.81%, and 8.27%, and increasing load balancing by 33.36%, 19.69%, and 59.06%, respectively.

In order to achieve better load balancing, a study [10] presented Load Balancing HEFT (LB-HEFT), that is a modification to the HEFT algorithm. LB-HEFT's performance was evaluated by comparing it to both the original HEFT algorithms and the E-HEFT. Outperforming both E-HEFT and HEFT, the results show that LB-HEFT considerably improves load balancing by 43.49% and 72.59% on average, improves resource utilization by 2.28% and 5.61% on average, and lowers Makespan by 7.55% and 3.75% on average. In order to achieve better load balancing, a study [11] presented Load Balancing HEFT (LB-HEFT), a modification to the HEFT algorithm. LB-HEFT's performance was evaluated by comparing it to both the original HEFT algorithms and the E-HEFT. Outperforming both E-HEFT and HEFT, the results show that LB-HEFT considerably improves load balancing by 43.49% and 72.59% on average, improves resource utilization by 2.28% and 5.61% on average, and lowers Makespan by 7.55% and 3.75% on average. In the study [12], job scheduling problems in cloud computing settings are addressed through metaheuristic and hybrid metaheuristic algorithms. We created metaheuristic algorithms by combining a greedy approach (GR) with genetic algorithms (GA), differential evolution (DE), and simulated annealing (SA).

Furthermore, we developed hybrid metaheuristic algorithms called GA-SA and DE-SA, which are also paired with a greedy strategy. These methods' virtual machine load balancing and completion time were used to evaluate them. According to the results, the average completion time of the DE-SA algorithm was higher than that of the separate DE and SA algorithms when the number of tasks increased. Additionally, tests showed that hybrid algorithms enhanced the average standard deviation of virtual machine loads and the average completion time for specific task groups. GWO and PSO, two well-known meta-heuristic algorithms, are combined to create the suggested algorithm known as PSO–GWO [13]. The experiment's findings demonstrate that, compared to the normal Particle Swarm Optimization and Grey Wolf Optimization algorithms, the PSO–GWO algorithm reduces the average total execution time and cost. The main research question we tackle in this work is the possible trade-off between the cost of using virtual machines and their Makespan [14]. We suggest using the ant colony algorithm (ACO) in conjunction with the heterogeneous earliest end time (HEFT) to minimize them. Three different real-world science workflows are used for experimental simulations, which also account for the characteristics of the Amazon EC2 cloud platform. The suggested method outperforms FR-MOS, PEFT-ACO, and basic ACO based on the experimental results.

The scheduling problem is solved by the hybrid algorithm described in a study [15] based on the MHPSLP mathematical model. It divides the problem into smaller subsets, such as scheduling task bags and allocating resources using the mixed integer linear mathematical (MILP) model. Compared to other scheduling algorithms, this method's advantage is a decrease in the cost of completed tasks under deadline constraints. The issue of energy consumption and effective resource usage in virtualized cloud data centers is discussed in the article [16]. The suggested technique is predicated on job classification and thresholds in order to improve resource consumption and scheduling effectiveness. Workflow tasks are preprocessed in the first phase by separating jobs with longer execution times and more dependencies into different queues in order to prevent bottlenecks. Tasks are categorized in the following phase according to the intensity of resources needed. Particle Swarm Optimization (PSO) is the final step in choosing the optimal schedules. Experiments validated the suggested method. Results from comparisons using benchmark datasets are shown. The outcomes demonstrate the suggested algorithm's superior efficacy over the other algorithms compared to in terms of makespan and load balancing. This publication[17] proposed a novel workflow-scheduling method that considers task priorities and efficiently schedules jobs onto matching virtual resources. The approach used to model this process was the whale optimization algorithm. Workflowsim simulator was used for extensive simulations. It was compared to the current PSO, CS, ACO, and GA algorithms. Ultimately, it was determined from the simulation findings that the energy consumption, Makespan, and migration time were all significantly reduced.

## 3. Proposed System

The proposed system for multi-objective task scheduling in a cloud environment begins by accepting input in the form of a scientific workflow dataset and a set of virtual machines (VMs) designated for task execution. This initial stage involves defining the tasks within the workflow and the available computational resources. Once the inputs are defined, the system calculates several critical metrics for each task on each VM. These metrics include Total Length of Task (TTL), which measures the cumulative length of the instructions of tasks assigned to each virtual machine (VM); Earliest Finish Time (EFT), which calculates the earliest time a task can be completed on a given VM; and VM-based Task Parent (TP), which indicates whether a task has a parent task on the same VM, taking into account the communication cost between related tasks. The proposed system design is shown in Figure 1.

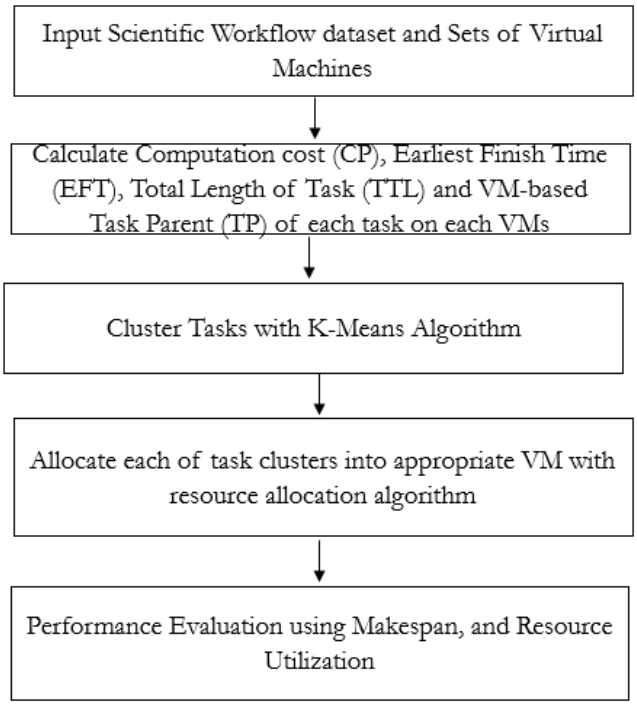


**Figure 1.** Proposed System Design

After calculating these metrics, the K-Means clustering algorithm is used to group tasks with similar characteristics. Clustering tasks based on their attributes allows for more efficient handling and allocation, as similar tasks are grouped, reducing the complexity of scheduling decisions and improving overall system efficiency. Each cluster represents a set of tasks with similar computational and communication requirements, which can then be managed collectively. In the next stage, the system utilizes a resource allocation algorithm to assign each cluster of tasks to the most appropriate VM. This allocation strategy considers the computational and communication requirements of the tasks within each cluster, ensuring that the selected VMs can handle the workload effectively. This step is crucial for optimizing the use of cloud resources and minimizing execution time.

Finally, the system evaluates performance using key metrics such as Makespan and Resource Utilization. Makespan refers to the total time required to complete the entire workflow, while Resource Utilization measures resource use efficiency. By optimizing these objectives, the system aims to enhance the overall performance and efficiency of task scheduling and resource allocation in the cloud environment. Integrating these stages ensures that the system can handle complex scientific workflows with optimal resource utilization, achieving high-performance computing outcomes.

Creating subsets within a dataset where items in the same group are more similar to each other than to those in different groups is the main goal of clustering. K-means clustering is a straightforward and effective method among the many clustering techniques accessible in data mining. K-Means is well-known for its speed and simplicity of use, and it is particularly effective at handling datasets with a single attribute compared to other data clustering algorithms. Consequently, to minimize Makespan and guarantee an equitable workload allocation among the virtual machines (VMs), the suggested method uses K-Means clustering to create task clusters.

"K" stands for the number of clusters that need to be defined beforehand in K-Means clustering. The procedure's first step is randomly choosing a centroid value for every cluster. After that, iteratively allocating jobs to the closest centroid, iteratively recalculates centroid values depending on the cluster members until convergence is achieved. This method efficiently groups tasks with comparable computational and communication properties. A balanced workload is achieved across all VMs by assigning these clustered tasks to appropriate VMs based on their processing capacity. By clustering tasks and considering VM capabilities, the system optimizes resource utilization, minimizes execution time, and achieves load balancing, enhancing overall performance in a cloud environment.

The suggested method optimizes resource allocation and work scheduling in a cloud environment by incorporating a number of essential characteristics. In Algorithm 1 (Task Clustering using K-Means Clustering), the variables are as follows: the input is a workflow of tasks, and the output is task clusters. The algorithm begins by selecting $k$ random instances as the centroids of the clusters. The number of clusters, $k$, is predefined, and the algorithm then computes the Euclidean distance ($d$) between each task $Ti$ and each centroid $Sj$. The distance formula is $d(T_i, S_j) = \sqrt{(Ti - Sj)^2}$, which measures how close a task is to a centroid. Each task Ti is assigned to the task cluster $TCj$ where the distance $d(T_i, S_j)$ is the smallest. After all tasks are assigned to clusters, the mean μ of all task lengths in each cluster $TCj$ is recalculated as the new centroid $Sj$. This process repeats until convergence, meaning the centroids no longer change or another stopping criterion is met.

In Algorithm 2 (Resource Allocation Algorithm), the variables include the input, which consists of task clusters and a list of VMs, and the output, which involves assigning tasks to clusters. Here, n is the number of task clusters, $z$ is the number of tasks, and m is the number of VMs. The algorithm iterates over each cluster $c$, each task $t$, and each VM $v$, aiming to assign tasks to VMs based on the same Euclidean distance formula used in clustering. This ensures that tasks are allocated to VMs in a manner that minimizes the difference between the task and the centroid, improving the efficiency of resource allocation in the cloud environment.

The Computation Cost (CP) of every task on every VM is the main emphasis of Feature 1. In order to ensure that the system can evaluate the processing demands and assign jobs to VMs that can manage the workload effectively, this feature computes the computational resources needed for each task. By understanding the computation cost, the system can make

more informed decisions about where to assign tasks, balancing the load and preventing resource overutilization or underutilization.

**Algorithm 1.** Task Clustering using K-Means Clustering Algorithm

INPUT: Workflow of tasks
OUTPUT: Task clusters

1: Select $k$ random instances as the centroids
2: Until clustering converges or other stopping criteria:
3: For each instance $Ti$
4: Find the Euclidean distance d between each $Ti$ and centroids $Sj$ of all the task clusters $d(T_i, S_j) = \sqrt{(Ti - Sj)^2}$
5: Assign Ti to the task cluster $TCj$ such that $d(Ti, Sj)$ is minimal
6: For each task cluster $TCj$, find the mean as $Sj = \mu(TCj)$ where μ is the mean of all task lengths in cluster $j$

**Algorithm 2. Resource Allocation Algorithm**

INPUT: Task Clusters and VMs List
OUTPUT: Task clusters

1: For $c = 1$ to $n$, Do // $n$ is the number of clusters
2: For $t = 1$ to $z$, Do // $z$ is the number of task
3: For $v = 1$ to $m$, Do // $m$ is the number of VMs
4: Find the Euclidean distance d between each $Ti$ and centroids $Sj$ of all the task clusters $d(T_i, S_j) = \sqrt{(Ti - Sj)^2}$
5: Assign Ti to the task cluster $TCj$ such that $d(T_i, S_j)$ is minimal
6: For each task cluster $TCj$, find the mean as $Sj = \mu(TCj)$ where μ is the mean of all task lengths in cluster $j$

Assigning the Task to the VM with the Earliest Finish Time (EFT) is Feature 2. This feature shortens the workflow's overall Makespan by prioritizing allocating jobs to VMs that can finish them the fastest. The system maximizes productivity and reduces waiting times by automatically choosing the VM with the quickest estimated completion time for incoming tasks. This guarantees that jobs are finished as soon as possible.

The third feature is about the Total Length of Tasks (TTL) Allotted to Each Virtual Machine. TTL stands for Task-to-Virtual Machine cumulative length of instructions. This feature ensures that no VM is overworked by helping to understand how the workload is distributed among the virtual machines. By monitoring and balancing the total length of tasks across all VMs, the system can achieve better load balancing, improving performance and resource utilization.

The idea of the VM-based Task Parent (TP), which determines whether a task has a parent task on the same VM, is introduced in Feature 4. This feature is represented by a binary value, where one denotes the presence of a parent task, and zero indicates no parent. It considers the communication cost between tasks, as tasks with parent-child relationships may require data exchange. By identifying and managing parent tasks, the system can optimize the scheduling process to minimize communication delays and costs, further enhancing overall efficiency.

These features collectively enable the system to manage task scheduling and resource allocation effectively, ensuring optimal performance, reduced Makespan, and balanced workload distribution in a cloud environment.

## 4. Results and Discussion

A simulator from the CloudSim toolkit was used to assess the performance of the suggested solution. A Windows 10 computer with 8GB of RAM was used for the trials. Four different virtual machine (VM) kinds and four different job types were taken into consideration for the experiment. The four types of Virtual Machines (VMs)—Small, MI (Memory Intensive), CI (Compute Intensive), and Large—are each designed to handle specific workloads based on their resource capacities. The Small VM is typically equipped with limited

processor power and memory, making it suitable for lightweight tasks like basic web hosting or small-scale applications. The Memory Intensive (MI) VM is designed for applications that require substantial memory but moderate processing power, such as large databases or in-memory analytics. The Compute Intensive (CI) VM offers high CPU performance, making it ideal for CPU-heavy tasks like scientific computations and real-time processing. Lastly, the Large VM combines both high processing power and memory, making it capable of handling complex, resource-demanding tasks such as enterprise applications and large-scale simulations. Each VM type ensures optimal performance by aligning resource allocation with the specific needs of the task, enhancing efficiency in cloud environments.

As shown in Table 1, the number of virtual machines (VMs) for each kind was fixed at 20, and the task lengths were randomly generated within a predetermined range. Table 2 provides the task parameter settings. To assess the system's performance, the number of tasks was varied from 100 to 500. The results were then compared with those obtained using existing algorithms, specifically Particle Swarm Optimization (PSO)[18], Cuckoo Search Optimization (CSO) [19], and Grey Wolf Optimizer (GWO)[20]. This comparison provides a benchmark for evaluating the efficiency and effectiveness of the new approach. In order to replicate a genuine cloud computing environment, two scenarios were included in the experiment: one with comparatively fewer jobs and one with a few high amounts of HADF tasks. This setup mirrors common conditions in cloud environments, where certain types of tasks dominate the workload, posing challenges for scheduling and resource allocation strategies.

In Table 1, MIPS stands for Million Instructions Per Second, which measures the computational speed of each VM, indicating how many instructions a VM can process in one second. RAM (GB) refers to Random Access Memory in gigabytes, which represents the memory capacity of each VM. The VMs are categorized as Small (S), MI, CI, and Large (L), where Small VMs have lower MIPS and RAM, while MI and CI VMs specialize in memory and computational power, respectively. Table 2 shows the task parameter settings, with the Length (MI) representing the task length in terms of Million Instructions. Tasks are categorized similarly to the VMs: S for small tasks (100–1000 MI), MI for memory-intensive tasks (1000–4000 MI), CI for compute-intensive tasks (8000–10000 MI), and L for large tasks (4000–10000 MI). These abbreviations streamline resource allocation and task scheduling discussions in cloud environments.

**Table 1.** Virtual machines with speed and memory.

| VM | Small | MI | CI | Large |
|---|---|---|---|---|
| MIPS | 1000 | 2000 | 8000 | 8000 |
| RAM (GB) | 1 | 8 | 2 | 8 |

**Table 2.** Parameter settings of tasks.

| VM | Small | MI | CI | Large |
|---|---|---|---|---|
| Length (MI) | 100-1000 | 1000-4000 | 8000-10000 | 4000-10000 |

The simulator generates tasks of various kinds within the given range using a random number generator. Ten percent of the jobs in scenario 1 were of the $Tjs$ type, and the remaining ten percent were of the $Tjc$, $Tjm$, and $Tjl$ types. To model the suggested approach, a few presumptions are:

1. Tasks and virtual machines (VMs) are diverse in character, with each task being independent and assigned to a single VM.
2. Presume that each job is assigned to a single virtual machine (VM) and that no task is interrupted while it is executed.
3. The resources needed for a task are always less than what is available.

This section assesses and contrasts the simulation results for measures like Makespan and resource utilization. The tasks are completed in batches of 100–500. Table 3 describes the parameter settings for this system. Figure 2 presents Makespan results and Figure 3 describes resource utilization for different algorithms.

**Table 3.** Parameter settings.

| Parameter | K-Means | PSO | GWO | CSO |
|---|---|---|---|---|
| Number of Clusters (K) | Dynamic, based on workload | N/A | N/A | N/A |
| Initialization Method | K-Means++ | Random Initialization | Random Initialization | Random Initialization |
| Distance Metric | Euclidean Distance | N/A | N/A | N/A |
| Max Iterations (K-Means) | 300 | N/A | N/A | N/A |
| Population Size | N/A | 30 | 30 | 30 |
| Max Iterations (Optimization) | N/A | 100 | 100 | 100 |
| Inertia Weight (w) | N/A | 0.7 | N/A | N/A |
| Cognitive Coefficient (c1) | N/A | 1.5 | N/A | N/A |
| Social Coefficient (c2) | N/A | 1.5 | N/A | N/A |
| Alpha, Beta, Delta | N/A | N/A | Dynamic (leaders) | N/A |
| Discovery Rate (pa) | N/A | N/A | N/A | 0.25 |
| Random Walk Probability | N/A | N/A | N/A | Based on Lévy flight |

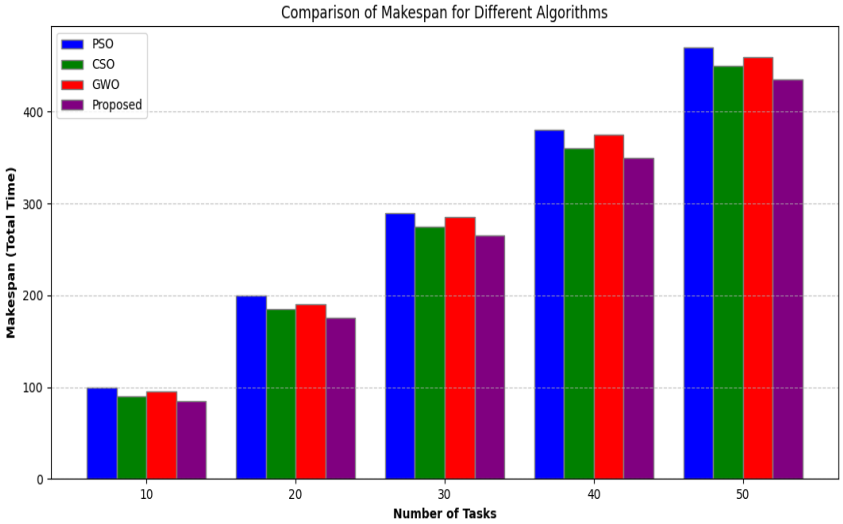


**Figure 2.** Makespan results

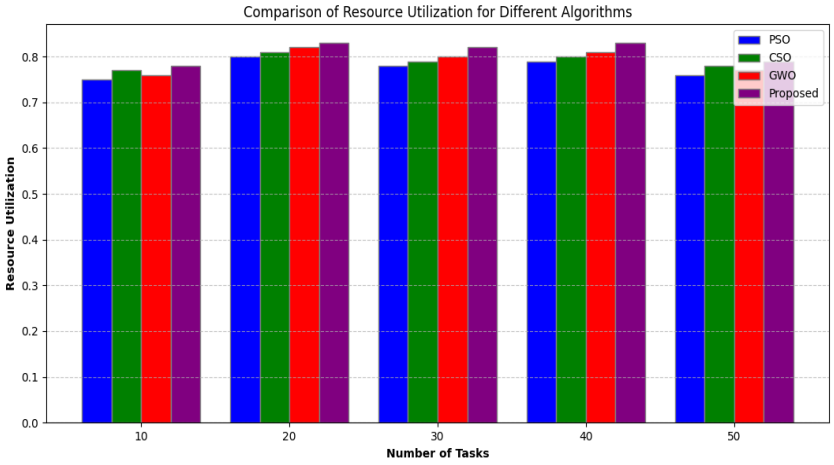


**Figure 3.** Resource Utilization

The makespan time determined by the suggested approach is consistently less than that of the other three algorithms as the number of tasks rises. The Makespan, which measures the total time required to complete all scheduled tasks, is significantly reduced in the proposed method when compared to traditional approaches like PSO, GWO, and CSO. This improvement is attributed to the K-Means clustering technique, which optimally groups tasks based on resource similarity, allowing for more efficient dynamic resource allocation. The Makespan is shortened by minimizing idle time and ensuring that tasks are assigned to the most appropriate resources, leading to faster task completion times.

The suggested strategy consistently yields a greater utilization value in the trial than the other three algorithms. Additionally, Resource Utilization is enhanced by the mul-ti-objective optimization process, which prioritizes not only minimizing Makespan but also maximizing the use of available computational resources. The proposed system ensures that VMs are allocated tasks in a balanced manner, avoiding overloading certain machines while under-utilizing others. This balanced allocation increases overall cloud system efficiency, re-duces task completion delays, and ensures optimal usage of CPU and memory resources across different VM types. Overall, the proposed method demonstrates clear advantages in both Makespan and Resource Utilization, optimizing task scheduling and resource allocation in a dynamic cloud environment.

The comparison of makespan and resource utilization clearly shows that the proposed method outperforms other algorithms (PSO, CSO, and GWO). For Makespan, the proposed method consistently delivers lower total time across all task numbers, with a makespan of just under 400 units at 50 tasks. In contrast, PSO exceeds 400 units, and both CSO and GWO approach 410 units. This suggests that the proposed method efficiently executes tasks faster due to its multi-objective optimization, which balances task allocation and resource usage better than the other algorithms. The proposed method's dynamic clustering using k-means also ensures optimal task grouping, minimizing delays. Regarding resource utilization, the method maintains consistently high utilization rates above 0.75, even with 50 tasks, while other algorithms like CSO and PSO fall below this threshold. For example, at 40 tasks, the proposed method achieves nearly 0.8 utilization, outperforming GWO and CSO. These results emphasize the method's effectiveness in minimizing Makespan and maximizing resource efficiency, making it ideal for environments that require both.

## 5. Conclusions

In this study, we have developed and demonstrated a comprehensive approach to multi-objective task scheduling for scientific workflows within cloud environments. Our methodology effectively addresses the complex challenges associated with optimizing computationally intensive scientific tasks by considering multiple conflicting objectives, such as execution time, resource utilization, and power consumption. By leveraging the scalability and flexibility of cloud resources, our approach dynamically provisions computing resources to match varying workloads, ensuring optimal performance. Integrating advanced metrics, including Communication Cost, Computation Cost, Earliest Finished Time on specific virtual machines (VMs), and Total Task Length, forms the foundation for informed decision-making in resource allocation. The use of the K-Means clustering algorithm to group similar tasks further enhances scheduling efficiency, while our specialized resource allocation algorithm ensures that tasks are assigned to suitable VMs based on their specific computational and communication needs by minimizing the Makespan, and maximizing resource utilization by comparing other task scheduling algorithms. This study emphasizes how crucial advanced scheduling methods are to maximizing cloud resource management for workflows in science. However, certain limitations were identified in the study. One of the key challenges is the scalability of the proposed method when dealing with extremely large datasets and high numbers of tasks. Although the method performs well in the tested scenarios, further optimization may be required for environments with more complex and larger-scale tasks. Additionally, the study focuses mainly on CPU and memory resources, while other aspects, such as network bandwidth and energy consumption, could also play significant roles in cloud task scheduling. Future work could explore these areas and further refine the approach to make it applicable in broader cloud computing contexts. In order to improve task scheduling and resource allocation even further, future research could investigate the integration of sophisticated machine-learning techniques and real-time modifications. To sum up, our mul-ti-objective task

scheduling approach offers a solid and scalable response to the difficulties associated with managing cloud resources, boosting the effectiveness and productivity of scientific study and hastening scientific advancements.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

[1] T. Bezdan, M. Zivkovic, N. Bacanin, I. Strumberger, E. Tuba, and M. Tuba, "Multi-objective task scheduling in cloud computing environment by hybridized bat algorithm," *J. Intell. Fuzzy Syst.*, vol. 42, no. 1, pp. 411–423, Dec. 2021, doi: 10.3233/JIFS-219200.

[2] A. Mohammadzadeh, M. Masdari, F. S. Gharehchopogh, and A. Jafarian, "Improved chaotic binary grey wolf optimization algorithm for workflow scheduling in green cloud computing," *Evol. Intell.*, vol. 14, no. 4, pp. 1997–2025, Dec. 2021, doi: 10.1007/s12065-020-00479-5.

[3] M. S. Sanaj and P. M. Joe Prathap, "Nature inspired chaotic squirrel search algorithm (CSSA) for multi objective task scheduling in an IAAS cloud computing atmosphere," *Eng. Sci. Technol. an Int. J.*, vol. 23, no. 4, pp. 891–902, Aug. 2020, doi: 10.1016/j.jestch.2019.11.002.

[4] M. Masdari, S. Gharehpasha, M. Ghobaei-Arani, and V. Ghasemi, "Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions," *Cluster Comput.*, vol. 23, no. 4, pp. 2533–2563, Dec. 2020, doi: 10.1007/s10586-019-03026-9.

[5] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A Firefly-based approach," *Appl. Soft Comput.*, vol. 93, p. 106411, Aug. 2020, doi: 10.1016/j.asoc.2020.106411.

[6] M. Farid, H. S. Lim, C. P. Lee, and R. Latip, "Scheduling Scientific Workflow in Multi-Cloud: A Multi-Objective Minimum Weight Optimization Decision-Making Approach," *Symmetry (Basel).*, vol. 15, no. 11, p. 2047, Nov. 2023, doi: 10.3390/sym15112047.

[7] M. Farid, R. Latip, M. Hussin, and N. Asilah Wati Abdul Hamid, "Weighted-adaptive Inertia Strategy for Multi-objective Scheduling in Multi-clouds," *Comput. Mater. Contin.*, vol. 72, no. 1, pp. 1529–1560, 2022, doi: 10.32604/cmc.2022.021410.

[8] M. Farid, R. Latip, M. Hussin, and N. A. W. Abdul Hamid, "Scheduling Scientific Workflow Using Multi-Objective Algorithm With Fuzzy Resource Utilization in Multi-Cloud Environment," *IEEE Access*, vol. 8, pp. 24309–24322, 2020, doi: 10.1109/ACCESS.2020.2970475.

[9] H. Mahmoud, M. Thabet, M. H. Khafagy, and F. A. Omara, "Multi-objective Task Scheduling in Cloud Environment Using Decision Tree Algorithm," *IEEE Access*, vol. 10, pp. 36140–36151, 2022, doi: 10.1109/ACCESS.2022.3163273.

[10] 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, Dec. 2021, doi: 10.1007/s10586-021-03334-z.

[11] S. Ghafir, M. A. Alam, F. Siddiqui, S. Naaz, S. S. Sohail, and D. Ø. Madsen, "Toward optimizing scientific workflow using multi-objective optimization in a cloud environment," *Cogent Eng.*, vol. 11, no. 1, Dec. 2024, doi: 10.1080/23311916.2023.2287303.

[12] M. N. Aktan and H. Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments," *Concurr. Comput. Pract. Exp.*, vol. 34, no. 9, Apr. 2022, doi: 10.1002/cpe.6513.

[13] N. Arora and R. K. Banyal, "A Particle Grey Wolf Hybrid Algorithm for Workflow Scheduling in Cloud Computing," *Wirel. Pers. Commun.*, vol. 122, no. 4, pp. 3313–3345, Feb. 2022, doi: 10.1007/s11277-021-09065-z.

[14] A. Belgacem and K. Beghdad-Bey, "Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost," *Cluster Comput.*, vol. 25, no. 1, pp. 579–595, Feb. 2022, doi: 10.1007/s10586-021-03432-y.

[15] M. Hariri, M. Nouri-Baygi, and S. Abrishami, "A hybrid algorithm for scheduling scientific workflows in IaaS cloud with deadline constraint," *J. Supercomput.*, vol. 78, no. 15, pp. 16975–16996, Oct. 2022, doi: 10.1007/s11227-022-04563-8.

[16] 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, p. 5849, Jun. 2021, doi: 10.3390/app11135849.

[17] S. Mangalampalli, G. R. Karri, and G. N. Satish, "Efficient Workflow Scheduling algorithm in cloud computing using Whale Optimization," *Procedia Comput. Sci.*, vol. 218, pp. 1936–1945, 2023, doi: 10.1016/j.procs.2023.01.170.

[18] G. Natesan and A. Chokkalingam, "Optimal Task Scheduling in the Cloud Environment using a Mean Grey Wolf Optimization Algorithm," *Int. J. Technol.*, vol. 10, no. 1, p. 126, Jan. 2019, doi: 10.14716/ijtech.v10i1.1972.

[19] K. Sreenu and S. Malempati, "FGMTS: Fractional grey wolf optimizer for multi-objective task scheduling strategy in cloud computing," *J. Intell. Fuzzy Syst.*, vol. 35, no. 1, pp. 831–844, Jul. 2018, doi: 10.3233/JIFS-17148.

[20] K. Sreenu and S. Malempati, "MFGMTS: Epsilon Constraint-Based Modified Fractional Grey Wolf Optimizer for Multi-Objective Task Scheduling in Cloud Computing," *IETE J. Res.*, vol. 65, no. 2, pp. 201–215, Mar. 2019, doi: 10.1080/03772063.2017.1409087.