INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

ORIGINAL ARTICLE

Cluster based Modified Minimum Completion Time Algorithm for Task Scheduling in Cloud Computing

N Kannammal[1]*, V Jude Nirmal[2], T Lucia Agnes Beena[3]

1 Research Scholar, Department of Computer Science, St.Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India
2 Assistant Professor & Research Supervisor, Department of Computer Science, St.Joseph's College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India
3 Assistant Professor, PG & Research Department of Computer Science, Holy Cross College (Autonomous), Affiliated to Bharathidasan University, Tiruchirappalli, 620 002, Tamil Nadu, India

OPEN ACCESS

Received: 30/06/2025
Accepted: 12/07/2025
Published: 22/08/2025

Citation: Kannammal N, Jude Nirmal V, Lucia Agnes Beena T (2025) Cluster based Modified Minimum Completion Time Algorithm for Task Scheduling in Cloud Computing. Indian Journal of Science and Technology 18(33): 2676-2689. https://doi.org/10.17485/IJST/v18i33.1118

* **Corresponding author**.

kannammal0506@gmail.com

Funding: None

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objective: To develop an efficient task scheduling algorithm in a cloud environment for reducing makespan, degree of imbalance and power consumption. **Method:** A novel Cluster-based Modified Minimum Completion Time (CMMCT) algorithm designed to enhance task scheduling in cloud environments. Initially, tasks are grouped based on their lengths using the Gaussian Mixture Model (GMM), which clusters tasks with similar computational demands. Following clustering, the Modified Minimum Completion Time (MCT) algorithm schedules tasks by evaluating both processing time and resource usage, ensuring optimal resource-task matching. The CloudSim framework is employed to evaluate the CMMCT algorithm against traditional scheduling techniques. The artificially generated dataset (comprising 200-1000 tasks and 40-200 VMs) is utilized to analyze the efficiency of the CMMCT. **Findings:** Results demonstrate substantial improvements in performance metrics such as Makespan, Total Execution Time, Degree of Imbalance, and Energy Consumption. These outcomes confirm the algorithm's ability to balance workloads efficiently while lowering scheduling overhead. Overall, CMMCT presents a robust and adaptive approach to task scheduling, enhancing both resource efficiency and service quality in cloud computing. **Novelty:** The traditional minimum completion time algorithm is modified with GMM cluster based tasks, and minimum time and power consumption factor is mainly used schedule the task. The suggested approach takes processing time and resource usage into account while selecting the best resource to schedule the task.

Keywords: Cloud Computing; Task Scheduling; Minimum Completion Time; Clustering; Heuristic Algorithm; Makespan; Total Execution Time; Energy Consumption

1 Introduction

Effective task scheduling methods are essential for maximizing efficiency across various computing resources[1]. Task scheduling can be classified as static or dynamic according on environmental factors[2]. In static scheduling, all tasks, resources, objectives, and limitations are predetermined before to the scheduler's execution. The scheduler subsequently assigns tasks to the appropriate resources at a designated time. In dynamic scheduling, the parameters of tasks and resources may fluctuate within the environment. This may affect the objectives and limitations during execution, rendering prior scheduling solutions suboptimal and, in certain instances, unfeasible[3]. Tasks may emerge, and resources can be added or eliminated during runtime. "To identify the optimal scheduling plan in either static or dynamic environments, various optimization techniques are commonly employed, including heuristic methods, meta-heuristic algorithms, and hybrid approaches that combine meta-heuristics with other techniques such as heuristics and machine learning."

"Heuristic algorithms[4] provide near-optimal solutions tailored to specific problems, while meta-heuristic approaches[5] are designed to deliver generalized solutions across diverse domains. Hybrid methods[6], which combine heuristics, meta-heuristics, or machine learning, are commonly used to tackle scheduling challenges in cloud computing. This study focuses on heuristic-based task scheduling techniques."

Nabi et al.[7] present a resource-aware dynamic task scheduling methodology. The method determines the computational allocation for each VM based on a series of tasks and thereafter selects a VM whose duration is either lower or equal to the computational allocation of the chosen VM. The method utilizes a load-balanced task allocation among available resources, so ensuring equitable use of all machines, resulting in enhanced resource usage and minimized execution time for task scheduling. Mubeen et al.[8] introduced an adaptive load-balanced job scheduling methodology for cloud computing. The task scheduling method allocates incoming tasks to accessible VMs in a load-balanced manner to decrease makespan, enhance resource usage, and adaptively reduce SLA violations.

Ding et al.[9] present a Q-learning-based framework for task scheduling aimed at enhancing energy efficiency in cloud computing. The M/M/S queue mechanism was built in the initial stage to allocate user requests for each node. During the next stage, a Q-learning-based scheduler at each node eliminates requests according to job laxity and deadlines. The jobs were allocated to the nodes followed by rewarding allocations that enhanced reaction time and resource consumption. However, energy consumption related to increased resource usage was not considered. Lavanya et al.[10] present two scheduling methods based on a threshold and service level agreement (SLA). The threshold-based task scheduling employed a prediction-driven methodology operating in two phases, utilizing threshold data derived from the ETC matrix, whereas the SLA proactively scheduled online jobs based on cost and deadline to decrease makespan and enhance resource utilization and load balancing.

Tong et al.[11] introduce the Double Deep Q-network (DDQN) for job scheduling. The adaptive learning capabilities of DDQN provided an optimal work schedule with shortest average reaction time and highest finish rate. The resulting degree of energy usage was disregarded in the study. Sharma et al.[12] present a novel method called priority-based load balancing (PLB) for cloud computing environments. It provides a resilient and adaptable job scheduling using multi-queues. This strategy enhances makespan, reaction time, resource utilization, and bandwidth by addressing the starving of low-priority jobs with available resources. The computational capacity of simulated virtual machines was not provided.

Yao et al.[13] offer a Hybrid Fault-Tolerant Scheduling Algorithm for independent jobs with specified deadlines. The drawback of this approach is to insufficient understanding of resource overhead resulting from fault tolerance. Krishnasamy et al.[14] present three heuristics for task scheduling: PTL (Pair-Task Threshold Limit), PTMax-Min, and PTMin-Max algorithms. There is no validation of performance in dynamic or large-scale situations.

Lipsa et al.[15] proposes a priority-based heuristic approach for task scheduling. This study presents a parallel approach for task scheduling, wherein priority assignment and heap construction are done concurrently, considering both non-preemptive and preemptive task characteristics. Thapliyal et al.[16] provides an effective work scheduling using the fuzzy logic (FL) with best-fit-decreasing (BFD) in a cloud computing environment. The approach is optimized for resource utilization, power consumption, cost, and time efficiency. Mustapha et al.[17] offer an approach utilizing DBSCAN (Density-Based Spatial Clustering) for task scheduling to attain enhanced efficiency. The DBSCAN-based task scheduling method seeks to enhance the quality of service for user tasks and optimize performance regarding execution time, mean start, and completion time. Most of the approaches have better results but few drawbacks: high computational complexity, high makespan and energy consumption. The comparative performance with conventional schedulers has not been examined.

Table 1 shows the summary of existing literature. It includes methodology, contribution, limitation or research gap.

This work presents an effective cluster-based modified minimum completion time (CMMCT) algorithm for task scheduling in cloud computing. In CMMCT, tasks are grouped according to task duration utilizing the Gaussian Mixture Model (GMM) clustering algorithm. The modified minimum completion time algorithm is employed for scheduling the task. The proposed method considers processing time and energy consumption while determining the optimal resource for task scheduling.

Table 1. Summary of Existing Literature

Ref.	Methodology	Contribution	Limitation / Research Gap
7	Resource-aware dynamic task scheduling methodology	Determines computational allocation for each VM and selects VM accordingly; refreshes VM load after iterations ensuring equitable usage.	Does not mention SLA violations or energy efficiency implications.
8	Adaptive load-balanced job scheduling methodology	Allocates tasks to VMs to minimize makespan, enhance resource utilization, and adaptively reduce SLA violations.	Detailed mechanism for SLA adaptation and energy efficiency not discussed.
9	Q-learning-based task scheduling framework	Employs M/M/S queue and Q-learning to enhance energy efficiency, job laxity, and deadline adherence.	Energy cost due to increased resource usage not considered.
10	Threshold and SLA-based task scheduling	Utilizes prediction-driven threshold and SLA scheduling for online jobs based on cost and deadline.	Effectiveness in real-time workloads or energy efficiency not evaluated.
11	Double Deep Q-Network (DDQN) for job scheduling	Uses DDQN for adaptive scheduling with minimal reaction time and high finish rate.	Energy consumption considerations omitted.
12	Priority-based Load Balancing (PLB)	Multi-queue job scheduling improves makespan, reaction time, resource utilization, and bandwidth.	Simulated VM computational capacity not provided.
13	Hybrid Fault-Tolerant Scheduling Algorithm	Chooses between resubmission and replication; online fault handling with dynamic elastic resource provisioning.	Limited insight on resource overhead due to fault tolerance.
14	PTL, PTMax-Min, and PTMin-Max heuristics	Uses task thresholds to sequence and group tasks for scheduling.	Performance in dynamic or large-scale environments not validated.
15	Priority-based heuristic with waiting time matrix and Fibonacci heap	Assigns task priorities using novel data structures; supports concurrent priority assignment and scheduling.	Computational complexity and scalability not addressed.
16	Fuzzy Logic with Best-Fit-Decreasing (BFD)	Optimizes scheduling for resource, cost, power, and time using fuzzy logic.	Uncertainty in real-time task characteristics and scalability not analyzed.
17	DBSCAN-based task scheduling	Applies clustering to enhance quality of service and performance in terms of execution, start, and completion time.	Comparative performance with traditional schedulers not explored.

The main contribution of this work is summarized as follows:

- A scheduling strategy is proposed to reduce Makespan, Total Execution Time, Degree of Imbalance and Energy usage.
- The Gaussian mixture model clustering is utilized to group tasks based to task length.
- The Cluster-Based Modified Minimum Completion Time (CMMCT) technique is proposed to enhance the efficiency of task scheduling.
- A simulation-based performance evaluation is conducted for the proposed scheduling technique.

2 Methodology

The Cluster-Based Modified Minimum Completion Time (CMMCT) method is an advanced task scheduling approach designed to optimize resource utilization and minimize execution time in cloud computing environments. It builds upon the traditional Minimum Completion Time (MCT) algorithm by introducing a clustering mechanism that groups similar tasks based on predefined criteria such as processing time, resource demand, or task priority. By organizing tasks into clusters, the CMMCT method enhances the decision-making process for resource allocation, ensuring that tasks are assigned to the most suitable virtual machines. This results in improved load balancing, reduced Makespan, and more efficient Energy Consumption. The method is particularly effective in heterogeneous cloud environments where workload diversity and dynamic resource availability pose significant scheduling challenges.

This section provides a comprehensive analysis of the proposed cluster-based modified minimum completion time (CMMCT) method for scheduling tasks. The suggested methodology has two phases: Clustering and Scheduling. In clustering, tasks are clustered according to task duration using the Gaussian mixture model (GMM) clustering algorithm. The modified

minimal completion time technique is employed for scheduling tasks. [Figure 1] illustrates the general architecture of the CMMCT.

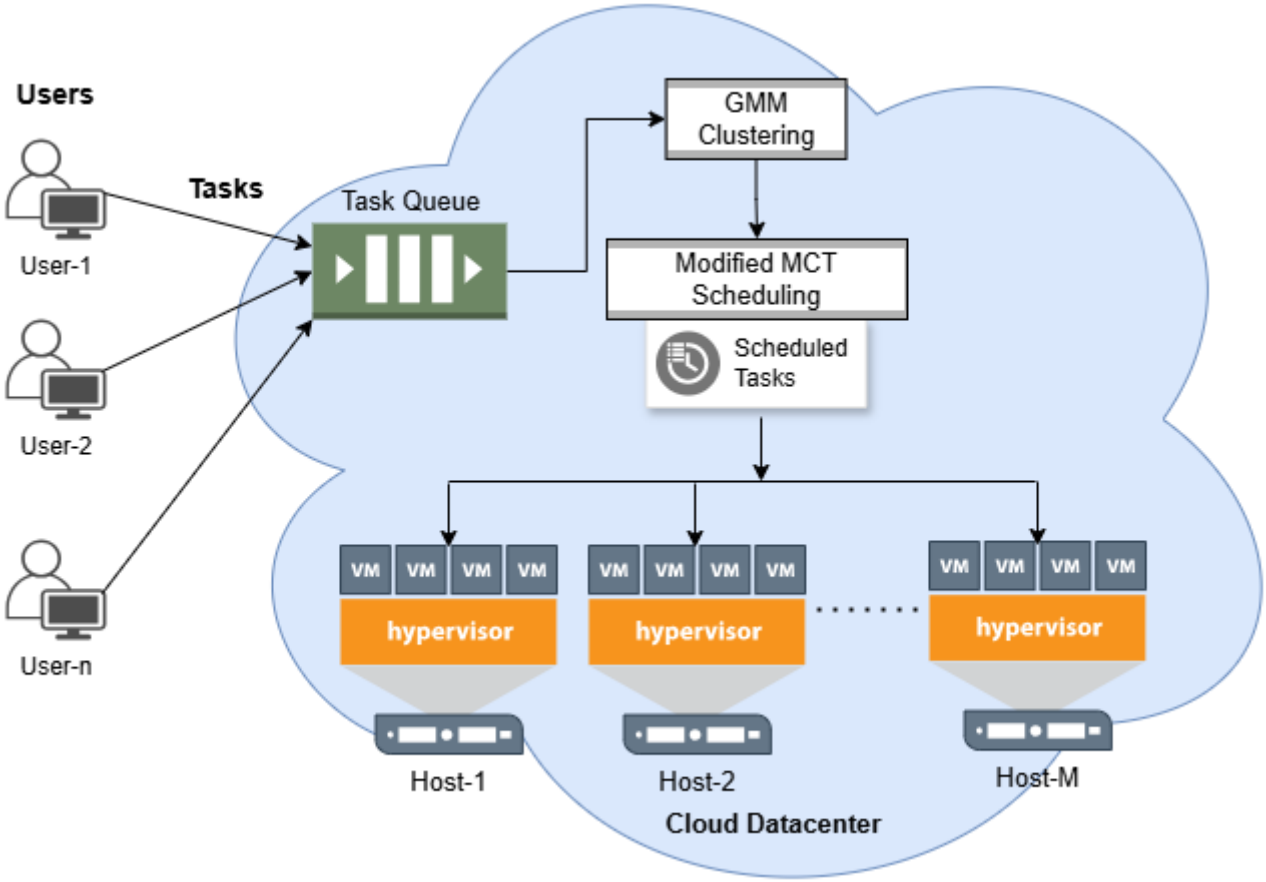

Fig 1. General Architecture of CMMCT Task Scheduling

The task scheduling problem in Cloud Computing can be precisely articulated as follows: Assign a collection of tasks T to a group of accessible virtual machines M in a cloud environment to meet the following objectives: decrease execution time and energy consumption, while maximizing resource usage.

In this paper, the task set T = $\{t_1, t_2, t_3, \ldots, t_n\}$ contains n number of tasks and each task is denoted by $t_i = \{tid_i, tlen_i\}$ where tid_i indicates the i^{th} task identification number and $tlen_i$ denotes the task length in Million Instructions Per Second (MIPS). The resource set VM =$\{vm_1, vm_2, vm_3, \ldots, vm_m\}$ comprises m virtual machines. VMs are varied physical computers with distinct computational capabilities. A virtual machine has several properties, including CPU processing speed expressed in million instructions per second (MIPS), memory, storage capacity, and bandwidth. Each VM in the resource set is denoted by $vm_j = \{vmid_j, MIPS_j, PCR_j\}$, where $vmid_j$ signifies the identification number of the j^{th} virtual machine. $MIPS_j$ denotes the processing speed, andPCR_j signifies the power consumption rate.

The Projected Computation Time (PCT) for processing the task i on VM j is denoted by,

$$PCT_{ij} = \frac{tlen_i}{MIPS_j} \tag{1}$$

The final result of the scheduling technique is an n x m distribution matrix indicating which task is assigned to each VM. The distribution matrix is defined as follows:

$$DisMat = \begin{bmatrix} \delta_{11} & \cdots & \delta_{1m} \\ \vdots & \ddots & \vdots \\ \delta_{n1} & \cdots & \delta_{nm} \end{bmatrix},$$

where

$$\delta_{ij} = \begin{cases} 1, & if\ t_i\ is\ assigned\ to\ vm_j \\ 0, & if\ t_j\ is\ assigned\ to\ vm_j \end{cases} \tag{2}$$

$$\sum_{j=0}^{m} \delta_{ij} = 1, \quad \forall\ 1 \leq i \leq n \tag{3}$$

The Execution Time (ET) of j[th] VM is computed using,

$$ET_{vm_j} = \sum_{i=0}^{n} \delta_{ij} \times PCT_{ij}, \qquad \forall\ 1 \leq j \leq m \tag{4}$$

The Energy Consumption(EC) of j[th] VM is computed using,

$$EC_{vm_{j=}} ET_{vm_j} \times PCR_j \tag{5}$$

2.1 Task Clustering

The Gaussian Mixture Model (GMM) clustering algorithm is a probabilistic model that assumes the data points are generated from a mixture of several Gaussian distributions with unknown parameters. It works by assigning data points to clusters based on the likelihood of their belonging to a specific Gaussian distribution in the mixture.

In this phase, Gaussian mixture model clustering algorithm is used to group the task based on the task length. A Gaussian Mixture Model (GMM) denotes a linear amalgamation of several Gaussian distribution functions. It is the most rapid learning probabilistic model[18].

The Gaussian distribution is a continuous probability represented by,

$$N(X|\mu,\sigma) = \frac{1}{(2\pi)^{\frac{D}{2}}\sqrt{|\sigma|}} \exp\left\{-\frac{(X-\mu)^{Tr}\sigma^{-1}(X-\mu)}{2}\right\} \tag{6}$$

where μ represents a D-dimensional mean vector, σ denotes a D $\times$ D covariance matrix that characterizes the Gaussian distribution, and $|\sigma|$ signifies the determinant of σ. The Gaussian distribution is symmetrical around the mean and is characterized by the mean and the standard deviation.

GMM seeks to identify a combination of multidimensional Gaussian distributions that most accurately represents the input dataset. GMM is an unsupervised clustering method that generates ellipsoidal groups based on probability density estimations utilizing the Expectation-Maximization algorithm. Every cluster is represented as a Gaussian distribution.

GMM is articulated as a linear amalgamation of fundamental Gaussian probability distributions and is formulated as,

$$p(X) = \sum_{k=1}^{K} \pi_k N(X|\mu_k,\sigma_k) \tag{7}$$

where K represents the number of elements in the mixture model, whereas π_k is the mixing coefficient, which provides a projection of the density for each Gaussian component. The Gaussian density represented by N(X|μ_k, σ_k) is referred to as an element of the mixture model. Each component k is characterized by a Gaussian distribution with mean μ_k, covariance σ_k, and mixing coefficient π_k.

Algorithm-1 explains the process of GMM clustering. The Gaussian parameters mean (μ_k), covariance (σ_k), and mixing coefficient (π_k) are initialized in step1. In step 2, each data point is allocated a score reflecting its probability of membership in each cluster. The assignment scores determine the degree to which data points belong to each cluster. The following equation is used to compute the score,

$$\gamma_{nk} = \frac{\pi_k N(x_n|\mu_n,\sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j,\sigma_j)} \tag{8}$$

The model parameters are revised according to the assignment scores acquired in step 2. Each parameter, from mean to covariance, is refined to more accurately align with the data distribution. The following equation is used to update the mean, covariance and mixing coefficients

$$\mu_k = \frac{\sum_{i=1}^{n} \gamma_{ik} X_i}{N_k} \tag{9}$$

$$\sigma_k = \frac{\sum_{i=1}^{n} \gamma_{ik} (x_i - \mu_k)^2}{N_k} \tag{10}$$

$$\pi_k = \frac{N_k}{n} \tag{11}$$

Where $N_k = \sum_{i=1}^{n} \gamma_{ik}$

The iterative process persists until the likelihood stabilizes, signifying convergence and denoting appropriate model parameters. If the convergence conditions are not satisfied, the procedure continues to iterate until stability is attained, ultimately yielding all model parameters for the Gaussian Mixture Model.

Algorithm-1: GMM Clustering
Input: Task Set T = {t_1, t_2, t_3, …, t_n}, number of cluster k
Output: Clustered Task CT = {C_1, C_2,…, C_k}
Step01: Randomly form initial cluster based on the task MIPS
Step02: For i = 1 to k
Step03: Compute initial mean $\mu_i = \frac{\sum x}{size(C_i)}$
Step04: Compute initial covariance $\sigma_i = \frac{\sum (x-\mu_i)^2}{size(C_j)}$
Step05: Compute initial mixing coefficient $\pi_i = \frac{n}{k}$
Step06: EndFor
Step07: For i = 1 to n
Step08: For j = 1 to k
Step09: Compute the rank γ_{ij}using Eq (8).
Step10: EndFor
Step11: Assign t_i to highest rank cluster
Step12: Update the mean, covariance and mixing coefficient using Eq. (9), (10) and (11)
Step13: EndFor
Step14: Return Clustered Data (μ_k, $\sigma_k$and π_k)

The clustered tasks are given to the task scheduling algorithm to efficiently allocate the tasks in VMs.

2.2 Proposed CMMCT Task Scheduling

The Minimum Completion Time (MCT) algorithm assigns tasks to virtual machines or resources based on the shortest expected completion time, processing them in a random order. Each task is mapped to the resource that can complete it the fastest. However, MCT may allocate some tasks to resources that do not offer the minimum execution time. To address this limitation, this paper employs a Modified Minimum Completion Time (MMCT) algorithm for improved task scheduling. Algorithm-2 explains the proposed CMMCT algorithm.

Algorithm-2 CMMCT Task Scheduling
Input: T = {t_1, t_2, …., t_n}, VM= {vm_1, vm_2, …., vm_m}, Clustered Task CT = {ct_1, ct_2, …, ct_k}
Output: Allocation of Task to VMs
Step01: For i = 1 to n
Step02: For j = 1 to m
Step03: Compute projected computation time PCT_{ij} using Eq. (1)
Step04: End For
Step05: End For
Step06: Initialize the elements of distribution matrix (DisMat) as zeros (Eq. (2))
Step07: For c = 1 to k
Step08: TempTasks = Get Task in cluster c
Step09: For i = 1 to Count(TempTasks)
Step10: MnT1 = Get the first minimum computation time of task i across all VMs
Step11: MnP1 = Get the first minimum power consumption of task i across all VMs
Step12: If (indexOf(MnT1) == indexOf (MnP1)
Step13: Allocate task i to VM at indexOf(MnT1)
Step14: Else
Step15: Allocate task i to VM at indexOf(MnP1)
Step16: End IF
Step17: Update the completion time of all VMs
Step18: Update DisMat
Step19: End For
Step20: EndFor
Step21: Return DisMat

In this algorithm, the projected computation time of i^{th} task and j^{th} is computed and initialize the elements of distribution matrix as zero. Compute the minimum computation time and power consumption of task for all VMs in each cluster. Allocate the task to VM based on the index. Update the completion time of all VMs and distribution matrix.

3 Results and Discussion

This section analyzes the performance metrics and presents the results of the proposed CMMCT methodology. Simulations are conducted using CloudSim, a toolkit designed for modeling cloud computing environments. The randomly generated tasks are used to analyze the performance of the proposed approach. The tasks and VMs range from 200 to 1000 and 40 to 200 is utilized for simulation. The performance of CMMCT is evaluated against other heuristic algorithms, such as MCT, Min-Min, and Max-Min [19], based on key criteria including makespan, total execution time, degree of imbalance, and power consumption.

The results are compared across various tasks and virtual machines to demonstrate the effectiveness of the proposed approach. Min-Min and Max-Min are chosen for task scheduling because they represent two complementary strategies:

- **Min-Min** focuses on minimizing response time and makespan for smaller tasks.
- **Max-Min** emphasizes fairness and balance, preventing large tasks from dominating execution time.

Together, analyzing both helps identify optimal task-to-resource mappings depending on workload characteristics and resource heterogeneity. Table 2 shows the simulation parameters.

Table 2. Experimental Setup

Parameters	Values
Number of Tasks	200 – 1000
Number of VMs	40 – 200
Task Length (MI)	1000 – 4000
VM (MIPS)	1000 – 5000
VM Power Consumption Rate (Watt)	200-1000

The proposed approach is evaluated using three metrics: makespan, power consumption and resource utilization. The makespan is defined as the highest processing time among all resources and it can be computed using,

$$Makespan = Max\left(ET_{vm_j}\right) \quad \forall\, 1 \le j \le m \tag{12}$$

Where ET_{vm_j} indicates the j[th] VM execution time.

Total execution time refers to the complete amount of time taken by a program or process to run from start to finish.

The total execution time (TET) is calculated as,

$$TET = \sum_{j=1}^{m} ET_{vm_j} \forall\, 1 \le j \le m \tag{13}$$

Degree of imbalance refers to the measure of uneven distribution of workload across multiple units, such as processors, tasks, or resources in a parallel or distributed system. It indicates how much one part of the system is doing more work compared to others.

The degree of imbalance (DoI) is computed using,

$$DoI = m * \left[\frac{Makespan - VM_{minET}}{\sum_{j=1}^{m}\sum_{i=1}^{n}\delta_{ij} * PCT_{ij}}\right] \tag{14}$$

Where VM_{minET} indicated the minimum execution time of every VM

The total power consumption is computed using,

$$TPC = \sum_{j=1}^{m} EC_{vm_j} \tag{15}$$

Where EC_{vm_j} represents the j[th] VM energy consumption

3.1 Evaluation Metrics

The performance evaluation results are derived from two sets of simulation circumstances. In scenario 1, the quantity of tasks ranges from 200 to 1000 (200, 400, 600, 800, 1000). The overall quantity of virtual computers is limited to 100.In scenario 2, the task count remains fixed at 500. The quantity of virtual machines ranges from 40 to 200 VMs (40, 80, 120, 160, 200).

Table 3 and Figure 2 show the performance of makespan comparison for fixed VM count and different number of tasks. From the results, when increasing the task count, the makespan also increased.

Table 3. Makespan Comparison (Fixed VM and Different No of Tasks)

Number of Tasks	MCT	Min-Min	Max-Min	CMMCT
200	2.2	2.7	2.4	**2.2**
400	4.2	6.6	5.9	**3.9**
600	9.4	10.7	9.9	**5.5**
800	12.1	16.0	14.9	**7.0**
1000	18.0	21.2	20.0	**8.6**

When compared to other algorithms, the conventional MCT and the proposed CMMCT have the shortest makespan for 200 tasks. The suggested CMMCT attained lowest makespan at all task counts. The Min-Min algorithm had the greatest makespan values across all different tasks.

From Figure 2, the proposed CMMCT reduces the makespan (1000 tasks and 100 VMs) for MCT, Min-Min and Max-Min approaches by 52%, 59.43% and 57% respectively.

Table 4 and Figure 3 show the performance of total execution time comparison for fixed VM count and different number of tasks. The total execution time increases when the task count is increased.

The proposed CMMCT algorithm's minimal overall execution time at 200 tasks is 152.8ms.

From Figure 3, the proposed CMMCT reduces the total execution time (1000 tasks and 500 VMs) for MCT, Min-Min and Max-Min approaches by14.97%, 21.35% and 19.39% respectively. The suggested methodCMMCT has a minimum execution time compared to other methods.

Table 5 and Figure 4 show the performance of DoI comparison for fixed VM count and different number of tasks.

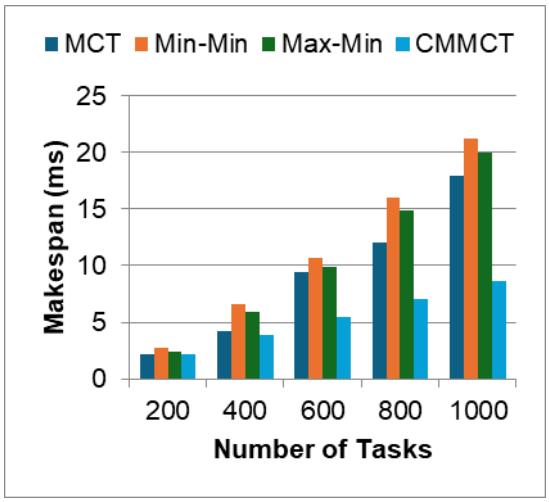

Fig 2. Comparison of Makespan for Fixed VM and Various Tasks

Table 4. Total Execution Time Comparison (Fixed VM and Different No of Tasks)

Number of Tasks	MCT	Min-Min	Max-Min	CMMCT
200	159.6	155.3	172.4	**152.8**
400	326.7	349.8	346.6	**315.3**
600	535.8	571.0	562.4	**500.9**
800	716.5	771.4	753.2	**671.2**
1000	972.3	1051.2	1025.6	**826.7**

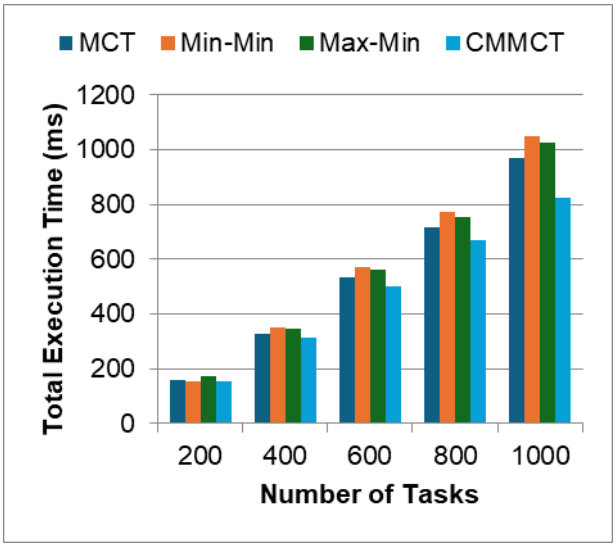

Fig 3. Comparison of Total Execution Time for Fixed VM and Various Tasks

Table 5. DoI Comparison (Fixed VM and Different No of Tasks)

Number of Tasks	MCT	Min-Min	Max-Min	CMMCT
200	0.9	1.7	0.6	**0.5**
400	0.6	1.3	1.0	**0.4**
600	1.1	1.3	1.1	**0.2**
800	1.1	1.5	1.4	**0.1**
1000	1.2	1.5	1.4	**0.1**

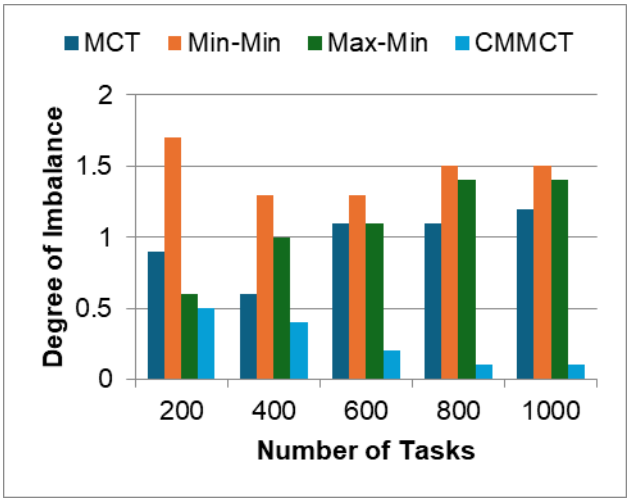

Fig 4. Comparison of DoI for Fixed VM and Various Tasks

From the results, the proposed CMMCT approach reduces the DoI of 0.5, 0.4, 0.2, 0.1, and 0.1 for 200, 400, 600, 800, and 1000 tasks respectively [Figure 4].

Table 6 and Figure 5 show the performance of energy consumption comparison for fixed VM count and different number of tasks. When increasing the task count, the energy consumption also increased. The proposed work reduces the energy consumption compared to all other algorithms.

Table 6. Energy Consumption Comparison (Fixed VM and Different No of Tasks)

Number of Tasks	MCT	Min-Min	Max-Min	CMMCT
200	123	141	130	**80**
400	242	341	314	**168**
600	476	527	496	**251**
800	600	749	709	**337**
1000	836	952	910	**415**

The proposed approach reduces the energy consumption for all different number of tasks. The Min-Min algorithm has highest energy consumption compared to all other algorithms.

From Figure 5, the proposed CMMCT reduces the energy consumption (1000 tasks and 100 VMs) for MCT, Min-Min and Max-Min approaches by50.35%, 56.40% and 54.39% respectively. The suggested methodCMMCT has a lower energy consumption compared to other methods.

Table 7 and Figure 6 show the comparison of makespan for fixed task count and different number of VMs. The makespan is anticipated to diminish with an increase in the number of virtual machines.

Table 7. Makespan Comparison (Fixed Task and Different No of VMs)

Number of VMs	MCT	Min-Min	Max-Min	CMMCT
40	25.9	27.5	25.1	**11**
80	10.6	14.8	11.8	**5.8**
120	5.5	7.8	6.1	**4.2**
160	3.7	4.9	4.1	**3.2**
200	2.7	3.7	3.2	**2.3**

The proposed CMMCT approach reduces the makespan for all different number of VMs compared to other algorithms. The Min-Min has highest makespan for all cases. The suggest work has 11, 5.8, 4.2, 3.2 and 2.3 ms makespan for 40, 80, 120,160, and 200 VMs.

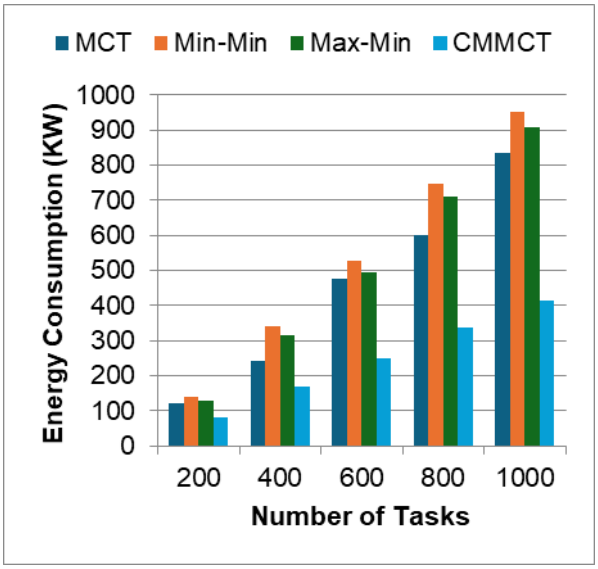

Fig 5. Comparison of Energy Consumption for Fixed VM and Various Tasks

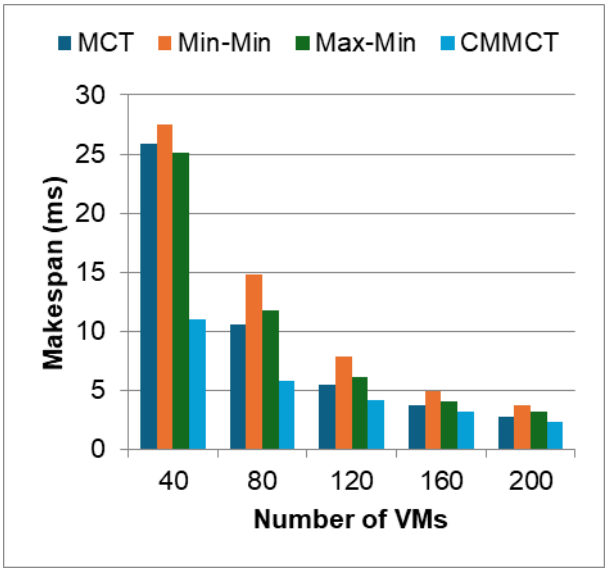

Fig 6. Comparison of Makespan for Fixed Task and Various VMs

From Figure 6, the proposed CMMCT reduces the makespan (500 tasks and 200 VMs) for MCT, Min-Min and Max-Min approaches by 14.81%, 37.83% and 28.12% respectively.

Table 8 and Figure 7 show the comparison of total execution time for fixed task count and different number of VMs. Similarly, the total execution time is anticipated to diminish with an increase in the number of virtual machines.

The proposed CMMCT approach reduces the execution time for all different number of VMs compared to other algorithms. The Min-Min and Max-Min has highest execution time for all cases.

From Figure 7, the proposed CMMCT reduces the total execution time (500 tasks and 200 VMs) for MCT, Min-Min and Max-Min approaches by 4.97%, 9.54% and 11.9% respectively.

Table 9 and Figure 8 show the comparison of DoI for fixed task count and different number of VMs.

The proposed approach reduces the DoI of 0.1, 0.2, 0.7, 0.6 and 0.5 for 40, 80, 120, 160 and 200 VMs respectively. The Min-Min algorithm has highest DoI for all the cases.

Table 8. Total Execution Time (Fixed Task and Different No of VMs)

Number of VMs	MCT	Min-Min	Max-Min	CMMCT
40	547	568.3	558.4	**419.7**
80	522	579.4	556.5	**434.5**
120	433.3	469.4	459.8	**411.2**
160	417.6	440.8	442.1	**399.8**
200	410.1	430.8	442.7	**389.7**

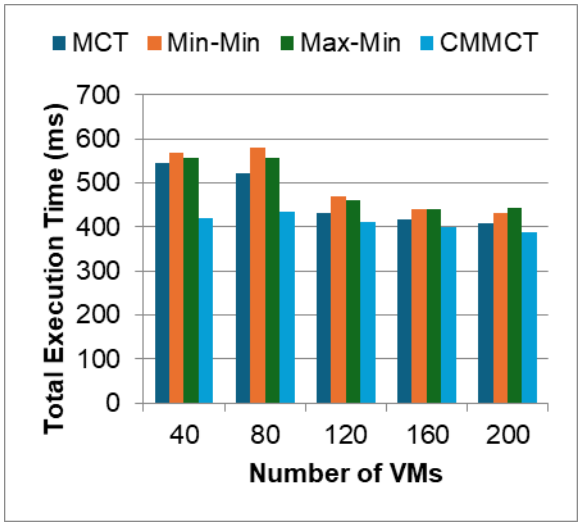

Fig 7. Comparison of Total execution time for Fixed Task and various VMs

Table 9. DoI Comparison (Fixed Task and Different No of VMs)

Number of VMs	MCT	Min-Min	Max-Min	CMMCT
40	1.4	1.5	1.3	**0.1**
80	0.9	1.6	1.1	**0.2**
120	0.8	1.4	0.9	**0.7**
160	0.7	1.2	0.7	**0.6**
200	0.7	1.7	0.6	**0.5**

Table 10 and Figure 9 show the comparison of energy consumption for fixed task count and different number of VMs. When increasing the VM count, the energy consumption is reduced.

Table 10. Energy Consumption Comparison (Fixed Task and Different No of VMs)

Number of VMs	MCT	Min-Min	Max-Min	CMMCT
40	461	482	449	**288**
80	392	511	426	**213**
120	340	444	367	**202**
160	318	395	344	**198**
200	304	375	336	**194**

The proposed approach reduces the energy consumption for all cases compared to other algorithms. The Min-Min algorithm has highest energy consumption compared to other algorithms.

Table 11 shows the performance comparison of metrics. The proposed approach CMMCT is compared with MCT, PSO, and MCT-PSO approach. The results reveal that the proposed methodology outperforms current approaches for makespan, total execution time, degree of imbalance, and energy usage.

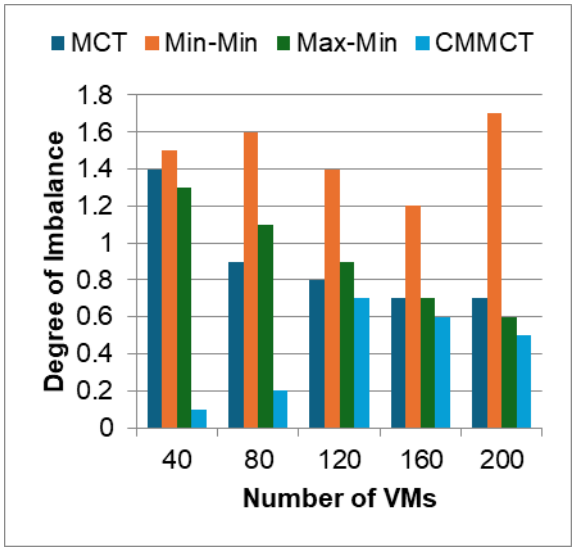

Fig 8. Comparison of DoI for Fixed Task and Various VMs

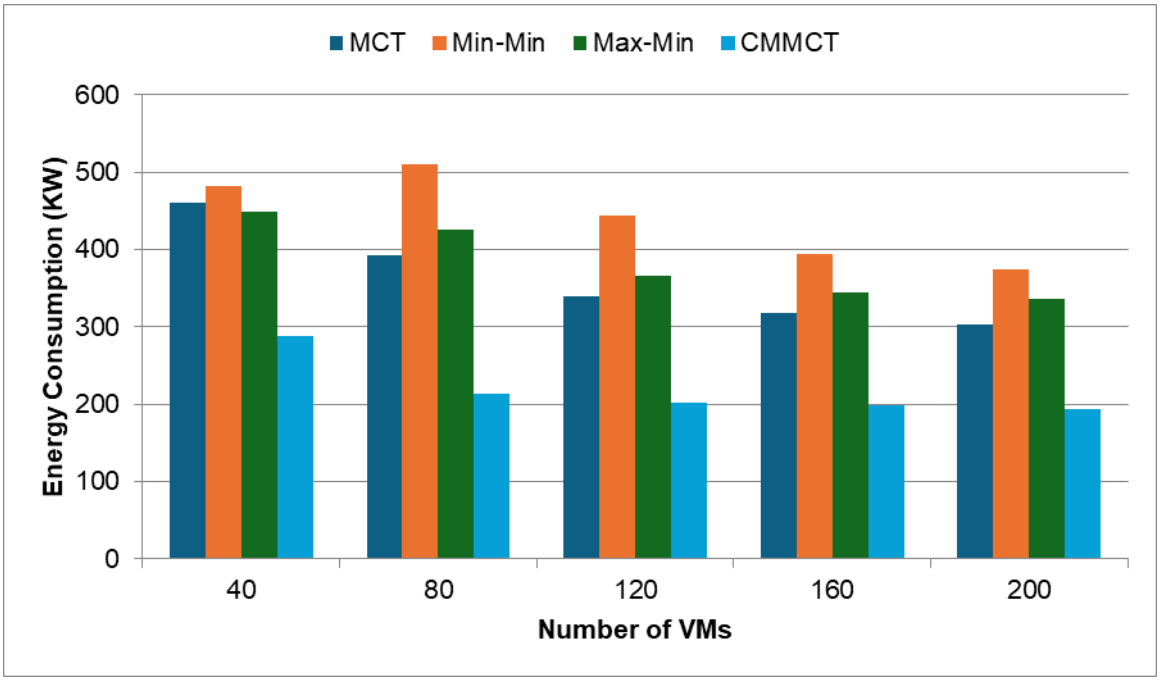

Fig 9. Comparison of Energy Consumption for Fixed Task and Various VMs

Table 11. Performance metrics comparison

Approach	Metrics			
	Makespan (ms)	Total Execution Time (ms)	DoI	Energy Consumption (kw)
MCT [19]	18	972.3	1.2	836
PSO [19]	20.7	931.1	1.8	935
MCT-PSO [19]	17.6	972.3	1.2	822
Proposed CMMCT	8.6	826.7	0.1	415

4 Conclusion

Efficient task scheduling constitutes a primary difficulty in cloud computing. The main purpose of this study is to create an efficient task scheduling algorithm in a cloud environment to minimize makespan, imbalance, and power consumption. This work proposes an efficient cluster-based modified minimum completion time algorithm for task scheduling in cloud computing environments. The conventional minimal completion time approach is adapted to incorporate GMM cluster-based tasks, primarily utilizing minimum time and power consumption factors to schedule the tasks. The proposed method considers processing time and resource utilization while determining the optimal resource for task scheduling. The CloudSim framework is utilized to assess the CMMCT algorithm in comparison to conventional scheduling methods. The suggested CMMCT decreases the makespan by 52%, 59.43%, and 57%, total execution time by 14.97%, 21.35%, and 19.39%, and energy consumption by 50.35%, 56.40%, and 54.39% in comparison to the MCT, Min-Min, and Max-Min methodologies. Comparative analyses reveal that the CMMCT approach outperforms traditional task scheduling algorithms in terms of makespan, total execution time, degree of imbalance, and energy consumption. The future scope includes:

- To use meta-heuristic algorithms to further enhance task scheduling efficiency
- To integrate supplementary performance metrics, including cost and resource utilization.
- To examine the application of several heuristic techniques for generating distinct initial search sites for metaheuristic algorithms.
- To implement a task scheduling algorithm in an adaptable cloud computing environment

References

1) Mahmoud H, Thabet M, Khafagy MH, Omara FA. Multiobjective task scheduling in cloud environment using decision tree algorithm. *IEEE Access*. 2022;10:36140–36151. https://doi.org/10.1109/ACCESS.2022.3163273.
2) Belgacem A, Beghdad-Bey K, Nacer H, Bouznad S. Efficient dynamic resource allocation method for cloud computing environment. *Cluster Comput*. 2020;23(4):2871–2889. https://doi.org/10.1007/s10586-020-03053-x.
3) Aburukba RO, Landolsi T, Omer D. A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices. *J Netw Comput Appl*. 2021;180:102994. https://doi.org/10.1016/j.jnca.2021.102994.
4) Mishra SK, Sahoo B, Parida PP. Load balancing in cloud computing: a big picture. *J King Saud Univ Comput Inf Sci*. 2020;32(2):149–158. https://doi.org/10.1016/j.jksuci.2018.01.003.
5) Houssein EH, Gad AG, Wazery YM, Suganthan PN. Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends. *Swarm Evol Comput*. 2021;62:100841. https://doi.org/10.1016/j.swevo.2021.100841.
6) Tanha M, Shirvani MH, Rahmani AM. A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments. *Neural Comput Appl*. 2021;33:16951–16984. https://doi.org/10.1007/s00521-021-06289-9.
7) Nabi S, Ibrahim M, Jimenez JM. DRALBA: Dynamic and resource aware load balanced scheduling approach for cloud computing. *IEEE Access*. 2021;9:61283–61297. https://doi.org/10.1109/ACCESS.2021.3074145.
8) Mubeen A, Ibrahim M, Bibi N, Baz M, Hamam H, Cheikhrouhou O. ALTS: An adaptive load balanced task scheduling approach for cloud computing. *Processes*. 2021;9(9):1514. https://doi.org/10.3390/pr9091514.
9) Ding D, Fan X, Zhao Y, Kang K, Yin Q, Zeng J. Q-learning based dynamic task scheduling for energy-efficient cloud computing. *Future Gener Comput Syst*. 2020;108:361–371. https://doi.org/10.1016/j.future.2020.02.018.
10) Lavanya M, Shanthi B, Saravanan S. Multi objective task scheduling algorithm based on SLA and processing time suitable for cloud environment. *Comput Commun*. 2020;151:183–195. https://doi.org/10.1016/j.comcom.2019.12.050.
11) Tong Z, Ye F, Liu B, Cai J, Mei J. DDQN-TS: A novel bi-objective intelligent scheduling algorithm in the cloud environment. *Neurocomputing*. 2021;455:419–430. https://doi.org/10.1016/j.neucom.2021.05.070.
12) Sharma G, Miglani N, Kumar A. PLB: a resilient and adaptive task scheduling scheme based on multi-queues for cloud environment. *Cluster Comput*. 2021;24(3):2615–2637. https://doi.org/10.1007/s10586-021-03280-w.
13) Yao G, Ren Q, Li X, Zhao S, Ruiz R. A hybrid fault-tolerant scheduling for deadline-constrained tasks in cloud systems. *IEEE Trans Serv Comput*. 2020;15(3):1371–1384. https://doi.org/10.1109/TSC.2020.2992928.
14) Krishnasamy KG, Periasamy S, Periasamy K, PrasannaMoorthy V, Thangavel G, Lamba R, et al. A pair-task heuristic for scheduling tasks in heterogeneous multi-cloud environment. *Wirel Pers Commun*. 2023;131(2):773–804. https://doi.org/10.1007/s11277-023-10454-9.
15) Lipsa S, Dash RK, Ivković N, Cengiz K. Task scheduling in cloud computing: A priority-based heuristic approach. *IEEE Access*. 2023;11:27111–27126. https://doi.org/10.1109/ACCESS.2023.3255781.
16) Thapliyal N, Dimri P. Task scheduling using fuzzy logic with best-fit-decreasing for cloud computing environment. *Cluster Comput*. 2024;27(6):7621–7636. https://doi.org/10.1007/s10586-024-04378-7.
17) Mustapha SD, Gupta P. DBSCAN inspired task scheduling algorithm for cloud infrastructure. *Internet Things Cyber Phys Syst*. 2024;4:32–39. https://doi.org/10.1016/j.iotcps.2023.07.001.
18) Patel E, Kushwaha DS. Clustering cloud workloads: K-means vs gaussian mixture model. *Procedia Comput Sci*. 2020;171:158–167. https://doi.org/10.1016/j.procs.2020.04.017.
19) Alsaidy SA, Abbood AD, Sahib MA. Heuristic initialization of PSO task scheduling algorithm in cloud computing. *J King Saud Univ Comput Inf Sci*. 2022;34(6):2370–2382. https://doi.org/10.1016/j.jksuci.2020.11.002.