KMPPVM-DRM: A KMeans++ Based Dynamic Clustering Approach for Virtual Machine Resource Allocation in Cloud Data Centers.

Mohanad Yahya Al-hamami[1,2], Mohsen Nickray[1]
[1]Department of Computer and Information Technology, Faculty of Engineering, University of Qom,Iran
[2]Information Technology Research and Development Centre, University of Kufa, Iraq
E-mail : mohanad.alhammami@uokufa.edu.iq, m.nickray@qom.ac.ir

Keywords: Cloud computing (CC), cloud data center (CDC), load balancing (LB), a virtual machine (VM), unsupervised learning clustering (ULC), resource allocation (RA), resource management (RM)

Received: March 30, 2025

Cloud computing(CC) delivers multiple services to users by processing complex tasks through internet connections to serve as a dynamic, dependable, and flexible computing solution. With the increased Internet speed, the user demands to perform many tasks through computing provided by cloud virtual machines (VMs) created in heterogeneous servers. The complexity of cloud computing(CC) increases through the diversity of servers and the increased demand for resources to perform more tasks, which makes the cloud data center(CDC) unbalanced in load. Therefore, future cloud systems will require more effective resource management(RM) methods to balance load and improve Quality of Service (QoS). This paper proposes KMPPVM-DRM, a dynamic clustering and scheduling approach for VM resource allocation(RA) based on the KMeans++ algorithm. VMs are clustered into three groups (high, medium, low) according to their real-time normalized CPU and RAM weights, using weighting parameters $\alpha = 0.6$ and $\beta = 0.4$. Tasks are categorized into three levels based on length and mapped to appropriate VM clusters. Within each cluster, a Utilization-Level Comparator (ULC) dynamically selects the optimal VM for task execution. The experimental results were conducted using CloudSim, with 5 and 10 VMs and task counts ranging from 1000 to 10000. The proposed model was compared with PSO, ACO, and DBSCAN algorithms using execution time, average start time, and average finish time. Results show that with 10 VMs, the proposed model achieved a harmonic mean execution time of 886.72 ms, compared to 4102.99 ms (PSO), 4672.25 ms (ACO), and 3071.51 ms (DBSCAN). It also attained the lowest harmonic mean start and finish times of 166.41 ms and 167.19 ms, respectively. Relative reduction in execution time against DBSCAN ranged from 69.90% to 71.52%, with start and finish time improvements between 52.65% and 81.51%.To ensure statistical reliability, a paired t-test confirmed that the performance improvements were statistically significant ($p < 0.05$) across all task sizes. The findings confirm that KMPPVM-DRM enhances resource allocation(RA) efficiency and scheduling effectiveness, maintaining balanced loads even with limited resources (e.g., only 5 VMs) and outperforming PSO, ACO, and DBSCAN in all tested scenarios.

Povzetek: Članek predstavi KMPPVM-DRM, pristop za razporejanje virov v oblačnih podatkovnih centrih. S pomočjo KMeans++ dinamično razvršča VM-je in izboljšuje porazdelitev nalog, zmanjšuje čas izvajanja ter povečuje učinkovitost oblačnih sistemov.

1 Introduction

Cloud computing (CC) enables users to access multiple services by processing Internet-based tasks while the cloud maintains a pay-per-use payment structure. CC is a computing model that delivers storage resources alongside devices and applications, extending the capability to scale distributed computing processes and task execution. The three cloud models offered by CC are infrastructure as a service (IaaS), platform as a service (PaaS), and software as a service (SaaS) [1]. The Infrastructure as a Service (IaaS) model is one of the most prominent trends in the world of (CC) to obtain an efficient and fast computing environment to perform advanced tasks. A Service Level Agreement (SLA) is signed between a Cloud Service Provider (CSP) (such as Amazon Web Service, Salesforce, etc.) and users who have a clear goal for the system with the required configuration in infrastructure as a service (IaaS) [2]. Figure 1 presents the different cloud service models adapted from [1].

Their requirements are formulated through QoS parameters, including (lowest cost, execution time, etc.). The cloud computing infrastructure contains cloud data centers consisting of large numbers of heterogeneous hosts through which virtualization technology is applied to carry out the tasks required in the computing process. The heterogeneous diversity of servers and the increased demand for resources to perform more tasks make the cloud data center(CDC) unbalanced in load [3]. The load balanc-

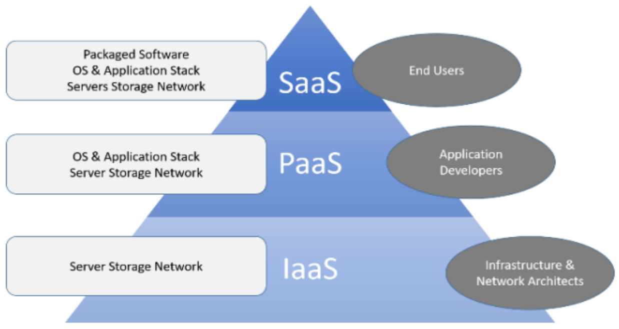


Figure 1: Presents the different cloud service models. source: [1]

ing technique improves the system's overall performance through high user satisfaction and resource utilization ratio while preventing any single node from becoming overwhelmed. Proper load balancing(LB) utilization results in optimal resource utilization, thus minimizing resource consumption [4]. The mechanism of LB in clouds enables a balanced distribution of dynamic workloads across cloud nodes. Load management raises the interest in efficiently balancing load with existing infrastructure, mainly through resource management(RM) through resource allocation(RA) and scheduling accomplished through policy-based algorithm implementation in the existing infrastructure [5]. Therefore, researchers have classified the existing LB algorithms into (static, dynamic, and hybrid). Static algorithms use already-known system information, including processing power, memory, performance data, and user requirements specifications [6]. The algorithms operate without requiring knowledge about the system's current state. A sudden failure of system resources and tasks creates significant problems for algorithms of this type. It is also inefficient, as user requirements cannot be constant in real time. Such as (Round Robin, Shortest Job Scheduling, Min-Min, Max-Min, (OLB + LBMM), CLBVM). [7] Dynamic algorithms make LB decisions based on the system state at a specific time because they operate without requiring any prior knowledge of the system conditions. The system drawbacks from static approaches can be eliminated through this method. The complexity of dynamic algorithms leads to enhanced performance in system operations. Some of them are (honey bee search, ant colony, biased random sampling, and evenly spread current implementation (ESCE)) [8]. Hybrid algorithms integrate two distinct algorithms to maximize the positive aspects of each procedure. Multiple researchers utilized machine learning technology by merging machine learning algorithms with load balancing algorithms for enhancing cloud performance; they include:

The research presents a hybrid GA-PSO algorithm that efficiently solves the task-to-RA problem [9]. The hybrid GA-ACO algorithm relies on utility-based scheduler output to determine optimization of task allocation through response time and throughput measurements with completion time calculations [10]. Researchers created A system by integrating modified Particle Swarm Optimization (MPSO) alongside an enhanced Q-learning algorithm named QMPSO [11]. The various LB algorithms enhance RM functions by effectively distributing resources. Cloud resource management requires VM allocation as an essential component that CDC uses to choose proper virtual machines which perform dependent or independent tasks [12] [13]. Service provider profits rose while average delays declined, and user satisfaction improved using TLBO (teaching-learning-based optimization) for dynamic RM and scheduling [14]. The research team developed an NSGA-II optimization framework, which operates as a multi-objective system to minimize service delay time and energy use under time restrictions to provide efficient scheduling and RM through optimal task distribution to achieve better load balancing [15]. The Current Resource Utilization drives periodic re-evaluations of existing VM allocation from available resources through dynamic VM allocation. However, in some scenarios, choosing too many virtual machines on a single host may lead to poor load balancing quality of service and thus increased SLA violations [16]. A system without intelligent resource scheduling faces significant complications in maintaining balanced workload execution without regard for the number of tasks present [17]. Despite developing various algorithms for resource allocation (RA) and load balancing (LB) in cloud computing, current methods such as DBSCAN, ACO, and PSO still face limitations in dealing with heterogeneous virtual environments. These approaches often struggle to effectively cluster virtual machines (VMs) based on real-time attributes such as CPU and memory availability and to assign tasks according to workload demands dynamically. As a result, they may suffer from increased execution time, inefficient resource utilization, and imbalanced load distribution across data center nodes, which can negatively impact the quality of service (QoS). This research identifies a performance gap in dynamic RA strategies, particularly the need for a more intelligent and adaptive method for grouping VMs and allocating tasks based on current system states. The study proposes a dynamic resource management (DRM) model named KMPPVM-DRM to address this gap. This model utilizes the KMeans++ clustering algorithm to dynamically categorize VMs into three groups (high, medium, and low) based on real-time CPU and RAM weights. Tasks are then classified according to their lengths and assigned to the most suitable VM within each cluster using a Utilization-Level Comparator (ULC). Accordingly, the study addresses key questions: Does using KMeans++ clustering help reduce the execution time of virtual machines compared to DBSCAN, ACO and PSO? Does dynamic task allocation based on the weight of VMs positively influence when average jobs start and finish? Can the new method contribute to better load balancing in cloud data centres? To achieve this, The main contribution of this research paper consists of the following points:

– Propose a dynamic weighting mechanism for VMs based on the characteristics of the current VMs, such as processing capacity and memory.

– Propose creating a clustering mechanism for virtual machines using machine learning (Unsupervised) that utilizes KMeans++ based on using the dynamically assigned weights.

– Propose a mathematical method for selecting the optimal virtual machine in each cluster.

– Construct a new approach named KMPPVM-DRM: "A KMeans++ - Based Dynamic Clustering Approach for Virtual Machine Resource Allocation in Cloud Data Centers", which dynamically clusters VMs and assigns tasks in a cloud computing environment to minimize execution time, the average start time of execution, and average finish while balancing the CDC load.

The proposed method, KMPPVM-DRM, is a hybrid approach to dynamic load balancing with unsupervised machine learning for organizing and scaling resources faster and more efficiently. It groups virtual machines (VMs) using weights to current details (CPU, RAM). Using KMeans++ for clustering, the system dynamically groups VMs and then applies a mathematical selection model to assign each task to the most suitable VM within its corresponding cluster. This strategy integrates the strengths of dynamic scheduling with intelligent VM grouping, aiming to reduce execution time, minimize task start and finish times, and achieve improved load balance and quality of service in cloud data centres. The paper divides its content into five sections, which include related work discussed in Section II and the proposed approach in Section III, followed by environmental evaluation with results in Section IV and the conclusion along with future work in Section V.

2 Related work

Managing load efficiency requires finding proper balances between present infrastructure and customer satisfaction through enhanced QoS [18]. The main objective of CC relies on managing resources and balancing loads through resource allocation(RA) and scheduling, which functions through its established policies and algorithms [19]. Which directly affects the quality of service (QoS). Many prior works have proposed various clustering and optimization techniques to enhance QoS and improve execution time. However, several methods lack adaptability to heterogeneous and real-time VM environments. This section presents an overview of key existing methods, identifies their strengths and weaknesses, and highlights the research gap that motivates our proposed KMPPVM-DRM model. Research studies in the literature focus extensively on this topic:

Q. Shang.[20] The author presented a dynamic RA algorithm built upon workflow and resource cluster approaches. The workflow represents precedence relations with subtasks and communication expenses through a directed acyclic graph model. The fuzzy clustering algorithm organizes nodes according to computing strength in addition to data transmission power and storage ability price levels and dependability levels. The cluster resources receive their assigned subtasks using a multi-objective optimization model. Reduce completion time and cost while improving resource utilization and load balancing.

S. El Motaki et al.[21] A clustering technique analyzed virtual machine conduct through resource information (CPU usage) and stall event details during workload execution in system environments. The proposed algorithm draws its essence from how birds naturally form logical clusters while in flight. The flight direction of each starling incorporates both self-sourced information and neighbor-starling-provided information. Each data item uses assigned weights to identify its subsequent feature space position after considering its existing location and nearby data values.

J.P.B. Mapetu et al.[22] presented a dynamic load-balancing framework that uses VM consolidation through four distribution methods. The proposed system implements three primary techniques for dynamic load balancing: VM selection according to imbalance level, BPSO for power optimization and host shutdown management, and Pearson correlation for maintaining SLA compliance. The proposed method decreases energy usage while diminishing SLA violations and decreasing VM migration frequency.

L. M. Al Qassem et al.[23] Two-state Random Forest (RF) machine learning models work proactively to predict future micro service workload requirements of CPU and memory resources. The predicted values from this process enable a dual adjustment of the resource pool by modifying hardware resources vertically while deploying additional micro-service replicas across horizontal axes.

G. Senthilkumar et al. [24] The authors developed a virtual machine allocation strategy that combines the approaches of the Random Forest (RF) and Genetic Algorithm (GA). Random Forest belongs to supervised machine learning techniques. This strategy makes Minimized energy usage possible by enabling better resource management(RM) for optimal utilization. A procedure exists for generating training data that enables random set training. The method is tested using real-time workload traces from Planet Lab. The proposed (GA-RF) model demonstrated better performance than other data center utilization techniques combined with host RM, power usage, and execution duration. The presented work uses resource usage, energy consumption, and execution time as performance metrics.

M. S. Al Reshan et al.[25] presented a combined GWO-PSO approach to improve system efficiency and RA in such a way that it takes advantage of the benefits of rapid convergence and global optimization. Research demonstrates that reaction times reached 12% below all competing algorithms in traditional methodology examination.

S. Singh et al. [26] presented a strategy, "Energy-Aware

Resource Allocation(RA) via MS-SLnO in Cloud Data," in which RA is achieved with the help of optimization to enhance the efficiency of cloud service. The method utilizes k-means clustering for task grouping. It deploys MS-SLnO as a hybrid bio-inspired optimizer that unites SLnO with MSA to find optimal resource allocations with PUE, which is included as a performance factor, execution time, and CPU usage factored into the decision. Through this approach, cloud data centers minimize their power consumption and reach maximum resource efficiency.

D. Kavitha et al.[27] Adaptive Deep Belief Network (ADBN) presents an adaptive deep belief network solution that incorporates Bayesian Search with Lichtenberg Optimization (BSI-LO) hybrid technique to address hardware limitations from resources, extended bandwidth delays, and slow performance of constrained devices. The offloading and RA methodologies achieved their best performance through this approach. The method provides an intelligent task offloading decision while achieving QoS and energy consumption balance.

Syed Mustapha et al.[28] proposed a method to enhance RA and task scheduling using (DBSCAN) algorithm, which is based on grouping virtual resources (VMs) according to their historical performance. This method reduced execution time and average start and finish time.

I. Jaya et al.[29] presented an adaptive deep reinforcement learning (DRL)-based method with adaptable capabilities deployed for edge-cloud gaming resource administration, reducing prices and enhancing allocation performance. The solution sought to achieve an equilibrium between cost expenditure and operational performance. The DRL-based algorithm reduces cumulative costs, brief allocation time, and large-scale performance capabilities.

H. Lin et al.[30] presented a two-level VM consolidation (TLHVMC) method that addresses the power mode transition (PMT) overhead requirements. The RA implemented a hybrid heuristic placement strategy based on a greedy strategy to achieve efficient resource placement (GSP). Time-based power mode control approaches are centralized management systems that control host power states. Wind power control strategies delivered higher energy efficiency through idle host suspension while using host state observations during a period to decrease SLA non-compliance.

Ahmad Raza Khan. [31] implemented dynamic load balancing through deep learning concepts, which unite CNNs and RNNs to derive load statistics for each VM. The clustering efficiency improves through the implementation of the Reinforcement Learning (RL) method together with the advanced Hybrid Lyrebird Falcon Optimization (HLFO) algorithm. The system dynamically divides VMs between overloaded and underloaded clusters, which leads to better resource use and reduced makespan durations.

O. K. J. Mohammad et al.[32] The researchers introduced Cloud Linear Regression (CLR) as their innovative machine learning framework, which integrates linear regression with cloud technology. CLR is a cloud-based enhancement that improves RM capabilities through better scheduling systems, provisioning, and allocation.

J. Ma et al.[33] The proposed study develops a Q-learning reinforcement learning method as a dynamic routing approach that adjusts network choices based on real-time conditions. The method decreases the root mean square error (RMSE) while enhancing load distribution to maximize resource usage.

C. Vijaya et al.[34] The proposed optimization method FCSFFC unites Cuckoo Search and Firefly Colony Optimization under fuzzy logic control for efficient cloud computing virtual machine migration. This technique improves RM and reduces superfluous migration instances for higher performance.

Table (1) below shows a summary comparison to highlight the main Proposed/Used Algorithm, metrics, results, strengths and weaknesses of the approaches.

From the comparative analysis in Table 1, key insights are observed : Most clustering-based methods (e.g.,[20],[21],[26],[28],[30]) are either static or rely on historical metrics or fuzzy clustering logic, which limits their real-time adaptability. For instance, fuzzy clustering[20] and bird-flocking-inspired clustering[21] yield good grouping but suffer from parameter uncertainty and poor adaptation to workload variation. Bio-inspired and deep learning strategies work well ([22]-[25],[27],[29],[31]-[34]) but often suffer from high resource demand, complexity, or convergence issues. Density-based techniques like DBSCAN([28]) struggle with high-dimensional resource heterogeneity and dynamic changes. Few works (e.g.,[26]) use simple K-Mean but lack justification for cluster initialization or adaptation to VM heterogeneity. Given the above limitations, a clear gap exists in developing a lightweight, adaptive, and resource-aware clustering mechanism that balances tasks in heterogeneous cloud environments. To address this, the proposed method(KMPPVM-DRM) introduces KMeans++, which improves KMean by using a stronger way to choose centroids, which results in both faster results and easier separation of clusters. This is especially beneficial for adaptive load balancing in heterogeneous VM environments based on real-time resource attributes (CPU, RAM).

3 The proposed approach

This section explains the proposed approach named A KMeans++ -Based Dynamic Clustering Approach for Virtual Machine Resource Allocation in Cloud Data Centers (KMPPVM-DRM) in the CC environment. The main objective of this approach focuses on delivering quality services to clients while decreasing total execution time, start times, and finishing times and establishing CDC workload equilibrium through resource allocation(RA) and scheduling mechanisms. The proposed is an advanced model based on machine learning algorithms for dynamic RA and scheduling that adapts to the heterogeneous CC environment. The proposed method (KMPPVM-DRM) consists of

Table 1: Comparative summary of related works

References, Year	Proposed/Used Algorithm	LB Metrics	Results	Strengths	Weaknesses
Q. Shang.[20], (2021)	Dynamic RA based on Workflow and Resource Clustering	Computing capability, cost, reliability	Effective in reducing completion time and resource utilization	Balanced allocation	Parameter uncertainty in fuzzy clustering leads to suboptimal results
S. El Motaki et al.[21], (2021)	Clustering Based on Bird Flocking Behavior	CPU usage, stall events	Effective clustering, improved resource usage	Fast, nature-inspired grouping	Reduced adaptation to transaction changes
J. P. B. Mapetu et al.[22], (2021)	Dynamic LB using VM Consolidation	Power, VM migrations, SLA	Reduced energy, SLA violations, VM migrations	SLA-aware, power-efficient	VM migration affects SLA and stability
L. M. Al Qassem et al.[23], (2023)	Proactive Autoscaling (Random Forest)	CPU, memory usage	Improved RA, reduced SLA violations	Predictive autoscaling	Historical-data based accuracy, slow response to workload changes
G. Senthilkumar et al.[24], (2023)	Hybrid Strategy (RF + GA)	Energy, resource usage	Reduced energy, improved utilization	Hybrid optimization	GA delays performance when optimization starts
M. S. Al Reshan et al.[25], (2023)	Combined GWO-PSO	Response time	Reduced response time by 12%	Fast convergence	Premature convergence complicates optimization
M. S. Al Reshan et al.[26], (2023)	Energy-Aware RA (MS-SLnO)	Power, CPU, execution time	Reduced power, optimized utilization	Hybrid optimizer	Requires calibration, affecting time and efficiency
D. Kavitha et al.[27], (2023)	Adaptive DBN (BSI-LO)	QoS, energy consumption	Improved QoS-energy balance	Intelligent task offloading	High resource demand, complexity, slow convergence
Syed Mustapha et al.[28], (2023)	DBSCAN-based Grouping	QoS	Improved QoS (13%), exec. time, start time (49%)	Effective historical grouping	Poor adaptation to dynamic changes
I. Jaya et al.[29], (2024)	Scalable DRL for Edge-Cloud Gaming	Cost, performance	Lower cost, high scalability	Smart dynamic learning	Needs large data and high computation
H. Lin et al.[30], (2024)	Two-Level VM Consolidation (GSP)	Power transitions, SLA	Energy savings, less SLA violations	Strategic placement	SLA violations due to PMT overhead
Ahmad Raza Khan.[31],(2024)	DL(CNNs, RNNs, RL)	Makespan, utilization	Improved utilization, lower makespan	Robust hybrid model	High resource/time cost of hybrid DL
O. K. J. Mohammad et al. [32],(2024)	Cloud Linear Regression (CLR)	CPU, Memory, Disk, Time	CPU: 59.8%, Mem: 72%, Disk: 90%, Time reduced	Smart VM selection, scalable	Limited to private settings, no benchmark
J. Ma et al. [33], (2025)	Cuckoo Search Algorithm (CA)	Time, Utilization, Scalability	Time: 2063ms, RR Time: 43088ms, LB: 95%	Scalable, global search	Needs tuning, poor under low resources
C. Vijaya et al.[34], (2024)	FCSFFC (Fuzzy Cuckoo + Firefly)	Energy, Migration, Computation	Improved energy, migration, load, computation	Fuzzy + bio-inspired exploration	Increased real-time complexity, needs fuzzy tuning

the following:

A. Initially, when tasks arrive at the broker, they are classified based on their length property, assuming a scenario with three different task length levels: high, medium, and low. Suitable virtual machines (VMs) are then allocated to them.

B. Assign initial weights to virtual machines based on their characteristics(CPU, RAM) from heterogeneous physical machines in a CC environment. The weights are calculated using the following Equation (1) :

$$VM_w = \frac{(\alpha * Cpu_{vm}) + (\beta * Ram_{vm})}{100} \quad (1)$$

The VM_w refers to the calculated VM weight, and both symbols(α, β) in Equation (1) represent the relative impact weights of the virtual machine characteristics. Where (α) is the relative impact weight of the CPU and the symbol (β) is memory (RAM) in the calculation.These values (α = 0.6) and (β = 0.4) were chosen to represent the higher significance of CPU than RAM in situations related to cloud computing. Also, (Cpu_{vm}) in Equation (1) refers to the CPU resources available and (Ram_{vm}) refers to the available RAM resources in the VMs. Weights for the virtual machines are stored in the weight map containing the VM number, the current available CPU, RAM, and the current machine weight. These weights are updated in real time. The calculated weight for the VM reflects the importance and amount of resources allocated based on the current (CPU, RAM) characteristics and the specified impact factors. These weights are used to help cluster the VMs.

C. Based on the weights assigned to virtual machines, these machines are clustered into several levels of availability (high, medium, low) so that the clustering process is done using (unsupervised) machine learning through the updated KMeans++ algorithm and based on real-time dynamic weight map data within the data center of the cloud computing infrastructure. Virtual machines are divided into k clusters in KMPPVM-DRM using KMeans++ as the clustering method. Here are the essential features of the algorithm:

- Number of Clusters (k): This study sets The number of clusters fixed to k = 3 since it reflects three distinct degrees of VM capability. These levels align with the classification of tasks(long, medium, short).

- Centroid initialization Strategy: Centroids are chosen for KMeans++ using the VM weights in a probabilistic fashion. Thanks to this method, the solutions have a better distribution and reach convergence faster.

- Distance Metric: The algorithm uses an Euclidean distance measurement based on the VM weight value, including CPU and RAM characteristics. It allows for good grouping according to how many resources are available.

Then, each virtual machine with the highest weight(i.e., the VM currently having the highest available resource capacity based on real-time CPU and RAM weighted characteristics) each clustering is assigned to the corresponding tasks, as defined in Equation (2) :

$$VM_{assigned} = argmax_{vm_i \in C_j} \left(W_{vmi}\right) \qquad (2)$$

Where ($VM_{assigned}$) the virtual machine selected for the next task. (VM_i) a virtual machine in the cluster C_j. (C_j) the cluster containing the virtual machines. W_{VMi} The weight of the virtual machine VM_i, representing its optimal capacity or suitability for the task. (argmax) The function that returns the argument (virtual machine) with the maximum weight. The selection approach ensures that the VM with the highest weight, reflecting its real-time available CPU and RAM capacities according to the specified impact factors (α and β), is prioritized for task execution. This helps to minimize execution time, avoid overloading less capable VMs, and load balancing in the heterogeneous cloud computing environment. Aftre the calculation of virtual machine weights basedon Equation(1) and the selection mechanism described in Equation(2),the notations and parameters involved in the proposed KMPPVM-DRM model are summarized fro clarity in Table 2.

D. The classified tasks are assigned to the optimal VMs from each cluster as follows:

   (a) The task with the most extended length is directed to the optimal VM in the cluster with a high weight level of currently available resources.

Table 2: Notation description of the proposed model

Symbol	Description
α	The relative impact weight of the CPU resource (assigned value 0.6)
β	The relative impact weight of the RAM resource (assigned value 0.4)
VM_w	The computed weight of a virtual machine
Cpu_{vm}	The CPU resource capacity of a virtual machine (in MIPS)
Ram_{vm}	The RAM resource capacity of a virtual machine (in MB)
$VM_{assigned}$	The virtual machine selected for the incoming task
VM_i	A specific virtual machine within cluster C_j
C_j	The cluster containing the virtual machines
W_{VMi}	The weight value of virtual machine VM_i
arg max	The function selecting the VM with the maximum weight value

   (b) The task with the medium length is assigned to the optimal machine in the cluster with the medium weight level of currently available resources.

   (c) The task with the shorter length is assigned to the optimal machine in the cluster with the low weight level of currently available resources.

As tasks assigned to the optimal VMs in each cluster start to execute, the weight map of the VMs is dynamically updated during the current process. The KMPPVM-DRM approach is applied based on the updated weights, and this process is repeated continuously until all the required tasks are completed, as shown in the following Figure (2). Overall, the proposed approach supports dynamic allocation and scheduling of resources to provide efficient load balancing in resource utilization and improve overall system performance.

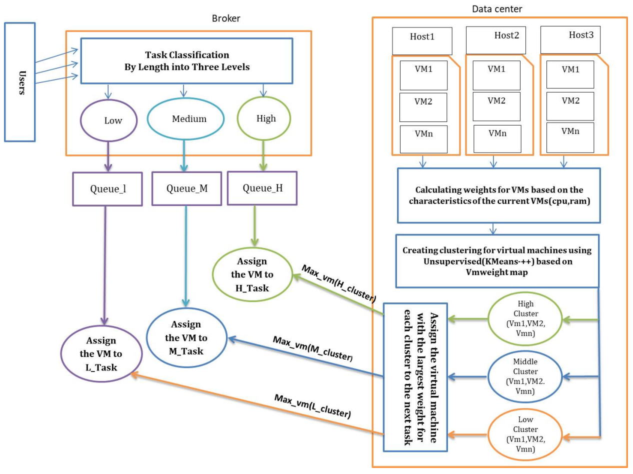

Broker
Data center
Users
Task Classification By Length into Three Levels
Low
Medium
High
Queue_l
Queue_M
Queue_H
Host1
Host2
Host3
VM1
VM2
VMn
Calculating weights for VMs based on the characteristics of the current VMs(cpu,ram)
Creating clustering for virtual machines using Unsupervised(KMeans-++) based on Vmweight map
Assign the VM to H_Task
Assign the VM to M_Task
Assign the VM to L_Task
Max_vm(H_cluster)
Max_vm(M_cluster)
Max_vm(L_cluster)
Assign the virtual machine with the largest weight for each cluster to the next task
High Cluster (Vm1,VM2, Vmn)
Middle Cluster (Vm1,VM2. Vmn)
Low Cluster (Vm1,VM2, Vmn)

Figure 2: Diagram of proposed model approach

Detailed steps in the proposed KMPPVM-DRM scheduling and RA method are shown in Figure 3 (Flowchart). At the start, each cloudlet is classified by its length into three categories: short, medium, and long. Then, the algorithm gives each virtual machine a dynamic weight based on the weighted values of both its CPU and RAM. All these weights are placed in a VM Weight Map to represent the current performance state of each VM. The map classifies the VMs into three categories (high, medium, and low) using the KMeans++ algorithm. For every cluster, the VM with the highest weight is assigned to deal with its group of cloudlets. Asks are then assigned accordingly to maximize resource utilization and efficiency. The algorithm for each part of the job calculates the cloudlet start time, finish time, and execution time. VM weights are updated during the scheduling cycle, considering both the state of the system and the results from execution. Once every task is done, the average quality of service metrics is calculated. Otherwise, the process iterates, adjusting the weights to keep the division of tasks responsive and balanced even as new demands appear.

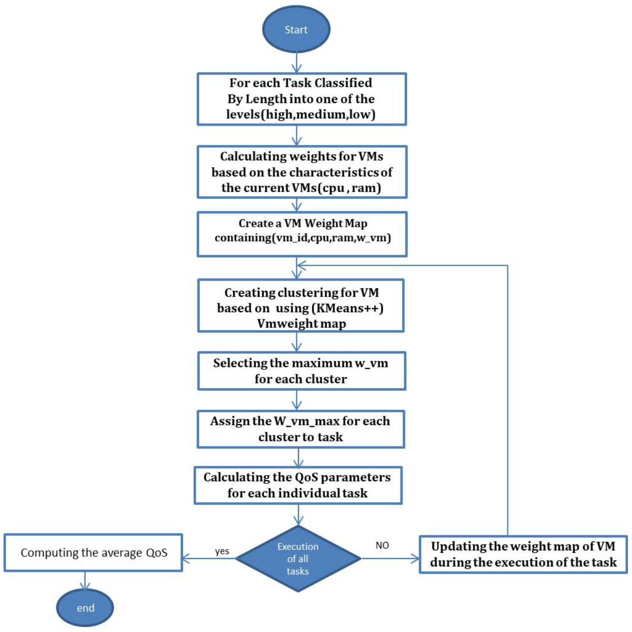

Start
For each Task Classified By Length into one of the levels(high,medium,low)
Calculating weights for VMs based on the characteristics of the current VMs(cpu , ram)
Create a VM Weight Map containing(vm_id,cpu,ram,w_vm)
Creating clustering for VM based on using (KMeans++) Vmweight map
Selecting the maximum w_vm for each cluster
Assign the W_vm_max for each cluster to task
Calculating the QoS parameters for each individual task
Execution of all tasks
yes
NO
Computing the average QoS
Updating the weight map of VM during the execution of the task
end

Figure 3: Flowchart of the proposed approach (KMPPVM-DRM)

The objective behind this approach consists of obtaining optimal resource allocation and performance enhancement for the system. The proposed KMPPVM-DRM approach contains a pseudocode to illustrate the formulas, parameters, and decisions involved in this approach.

The proposed KMPPVM-DRM approach was tested on two scales (e.g., 5 and 10 VMs and from 1000 to 10000 tasks). Although the current experiments are limited to these ranges, the clustering and weight-based selection mechanisms are designed to support scalability. KMeans++ adjusts to the number of virtual machines because it is an unsupervised algorithm.

Algorithm 1 The proposed approach (KMPPVM-DRM)

```
Input:
  – List of Cloudlets (Tasks)
  – List of VMs with characteristics
  – Clustering parameter $K = 3$
Output: Cloudlets assigned to VMs, updated VM resource data, clustering results
foreach cloudlet $t_i$ in TaskList do
    Classify $t_i$ into {High, Medium, Low} based on task length
while not all tasks are executed do
    foreach VM $v_j$ in VMList do
        Compute VM weight: $VM\_Weight_j = (\alpha \cdot CPU_j + \beta \cdot RAM_j)/100$ Store $VM\_Weight_j$ in $VMWeightMap$
    Apply KMeans++ clustering on $VMWeightMap$ to produce 3 clusters $C_1, C_2, C_3$
    foreach cluster $C_k$ do
        Select VM with highest weight: $VM_{\max} = \arg\max_{v \in C_k} W(v)$ Store $VM_{\max}$ in $VMmaxList$
    foreach task $t_i$ do
        Assign $t_i$ to appropriate $VM_{\max}$ based on task level (High, Medium, Low)
    Execute assigned tasks on selected VMs
    foreach executed task $t_i$ do
        Record execution time,start and finish time
    if all tasks completed then
        Compute and output average QoS(start and finish time) metrics break
    else
        Update VM weights dynamically to reflect current loads continue
```

The weight calculation formula and dynamic updating allow the method to extend to larger numbers of VMs and handle varied task lengths or multiple user requests. Future work will involve stress testing with more VMs (e.g., 50–100), the tasks, and concurrent user simulations. Also, The KMPPVM-DRM model's primary emphasis is on clustering for managing resources, while studies such as [32],[33], and [34] emphasize that Cloud Linear Regression (CLR) can help estimate resources and decide on tasks. The methods mentioned above are not part of this model, but we also see future work that combining clustering and regression might help allocate VMs more efficiently.

4 Environment and results

This section presents the experimental testing environment of the proposed model and provides a comprehensive analysis of the obtained results.

4.1 Experimental environment

4.1.1 Hardware configuration

The experiments were conducted on a personal computer with the processor, memory, and operating system specifications shown in Table(3). The purpose of such a configuration is to demonstrate the lightweight and cost-effective nature of the method, which does not require high-performance computing resources to achieve efficient results. Although advanced load balancing techniques such as deep learning often require high-performance GPUs and extensive memory.

Table 3: Hardware requirements

Component	Specification
Operating System	Windows(X64 based Processor)64-bit OS
Processor	Intel® Core™ i7-CPU@2.6 GHz
RAM	16.0 GB

4.1.2 Software and tools

Researchers who need to evaluate cloud performance and model cloud solutions use CloudSim as their primary simulation tool to avoid reliance on computing facilities and reduce costs. CloudSim is an importable tool that functions within programming platforms, including Eclipse and NetBeans IDE, and it enables Windows 10 users to create cloud simulations. Entities and computing resources are modelled by default to reflect the allocation and scheduling scenario in a cloud environment to evaluate the proposed method. The simulation is performed on Cloudsim 3.0.3. In this work, CloudSim version 3.0.3 was selected Because of its good performance stability, the ability to use Java-based add-ons, and the ability to use external Java libraries for clustering (for example, Smile). It suits the proposed model, ensuring compatibility with machine learning and reproducibility across similar research. A single data center with 2 hosts each and 5 and 10 virtual machines is used for the simulation. Table 4 lists the configurations of all the submitted tasks (cloudlets), which are separated by length into three groups. The values for these simulations were picked to show the same level of diversity found in other cloud task scheduling studies, for instance,[28]. Specifically, Short tasks (300) represent lightweight user requests. Medium tasks (2000–3000) reflect average processing needs. Long tasks (4000) simulate compute-intensive applications such as data analytics. Thanks to these classifications, we can see how the evaluation of load-balancing algorithms progresses under mixed workloads with varying execution complexities.

Table 4: Parameters of user tasks

Parameters	Values
Task length	300 , 2000/3000, 4000
Input File Size	200 Byte
Output File Size	400 Byte
PE	1-2

Table 5 lists Small, Medium and Large VM categories and their MIPS, RAM and PEs. These configuration values are adopted from Syed Mustapha et al.[28] to maintain consistency with existing benchmarks. The proposed KMPPVM-DRM method makes use of these configurations. In particular, the method splits VMs into low, medium and high resource categories using KMeans++, which are based on each VM's weight for CPU and memory. Because of this classification, the cloud environment can assign work by size and required resources, leading to balanced utilization of all resources in real-time.

Table 5: Type of VMs

	Mips, Ram(MB), Pe	VM category
VMs	(1000,512,1)	Small
	(2000,1024,2)	Medium
	(4000,2156,4)	Large

4.2 Performance metrics

For evaluating the proposed KMPPVM-DRM, this paper relies on different performance metrics to measure its performance through these specific evaluation metrics:

A) Execution Time (ExT): The time needed to perform tasks within a cloud environment on virtual machines (VMs) constitutes Execution Time (ExT) [35]. The VM performance relies on this metric as a primary computation measure because performance improvement requires lowering this metric value. It uses the approach outlined in [28] to estimate the execution time of a task as described by Equation (3) in this formulation.

$$ExT_i = \frac{Tasklength(i)}{NO.of Processor * VM(MIPS(j))} + NetworkDelayi \quad (3)$$

$Tasklength(i)$ indicates the number of instructions to be executed for $Task_i$ while while (No.of Processor) represents the number of processors(PEs) allocated for each task, $VM(MIPS_j)$ measures the processing speed of VM_j expressed in MIPS – Million Instructions Per Second, and $NetworkDelay_i$ represents all possible system delays, including dependency, resource, and data transfer times.

B) Start Time (ST) and Finish Time (FT): In cloud computing, a task's start time and finish time refer to when the task starts executing and the execution process is completed, respectively. These points are essential for scheduling, LB, and RA. ST is when a resource (such as a virtual machine or container) is allocated to the task and begins executing. (FT) The moment at which the task completes its execution, including any data transfer, computation, or storage operations. The (FT) or completion time(CT) is measured using the Equation (4) below:

$$FT_i = ST_i + ExT_i + D_i \quad (4)$$

ST_i stands for start time, ExT_i stands for execution time, whereas D_i represents any delay caused by dependencies, resource queuing, and data transfer. All measurements are expressed in milliseconds(ms).

In this context, 'improvement' reduces the fall of these metrics: ExT, Average ST, and FT. Lower values for all these metrics mean it operates more quickly, experiences fewer delays, and uses its resources better. Because of these improvements, using load balancing, QoS in the cloud is now more efficient.

4.3 Overall performance of the proposed model by varying the task count and VMs

In this section, the proposed (KMPPVM-DRM) algorithm performs evaluations against two different test cases: five VMs containing (1000 to 10000) cloudlets. Second: ten VMs containing (1000 to 10000) cloudlets. Increasing these variables can simulate reality and enhance the process of allocation and scheduling between different virtual machines for different cloudlets. The comparative analysis evaluated the proposed KMPPVM-DRM model against traditional and state-of-the-art algorithms.ACO[36] and PSO[37] acted as representative traditional metaheuristic optimization techniques because these two algorithms show broad application in classical resource scheduling problems. On the other hand, DBSCAN [28], a density-based clustering algorithm, was chosen for its recognized status as a state-of-the-art method in unsupervised learning due to its robustness in detecting clusters of arbitrary shapes and handling noise. This paper relies on performance metrics to measure its performance through these specific evaluation metrics:

1. Execution Time (ExT): Table 6 compares the performance of different models (PSO, ACO, DBSCAN, and KMPPVM-DRM) across various task counts and VMs. In the PSO model, execution time increases from 1478.47 (1000 tasks) to 13958.07(10000 tasks) and 10 VMs, indicating a proportional rise in execution time with an increasing task load. The ACO and DBSCAN follow similar trends, with execution time values escalating as the number of tasks grows. Notably, the proposed KMPPVM-DRM model stands out with significantly lower execution time values across all task counts (e.g., 307.75 for 1000 tasks), showcasing its superior efficiency in task execution time compared to the other models. Figure(4) shows the execution time of KMPPVM-DRM compared to the other models. The KMPPVM-DRM model's lower execution time suggests its potential for optimizing task scheduling and RA, resulting in faster task execution time and improved overall system efficiency.

Table 6: Shows execution time results while simulating 10 VMs measured in ms

Num of Tasks	PSO	ACO	DBSCAN	KMPPVM-DRM
1000	1478.47	1705.68	1080.168	307.75
3000	4177.67	4566.48	3240.378	975.25
5000	6960.87	8038.48	5400.624	1575.24
8000	11418.47	12774.48	8640.978	2475.24
10000	13958.07	16076.88	10801.22	3075.25

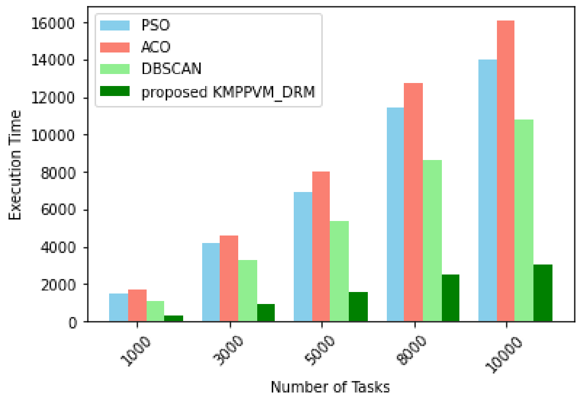


Figure 4: Execution time of KMPPVM-DRM compared to PSO, ACO and DBSCAN with 10 VMs

2. Finish Time : The average finish time values in the tables(7,8) are the average of moments at which the task completes its execution by various models (PSO, ACO, DBSCAN, and KMPPVM-DRM) in executing tasks across different counts and VMs. For PSO, the ACO, and the DBSCAN, there is a discernible upward trend in average finish time as the task count increases, indicating an increased finish time for handling larger workload scenarios, as shown in Figure (5,6). Specifically, DBSCAN's average finish time rises from 313.49 for 1000 tasks to 3103.79 for 10000 tasks with 5 VMs, rises from 219.44 for 1000 tasks to 2172.65 for 10000 tasks with 10 VMs. In contrast, the proposed KMPPVM-DRM model exhibits lower average finish time values across all task counts with VMs (5,10 VMs), underscoring its potential for energy-efficient task execution even with limited resources. For instance, KMPPVM-DRM achieves a notably lower average finish time of 123.24 for 1000 tasks with 5VMs and 61.93 for 1000 tasks with 10 VMs, positioning it as a promising model for scenarios prioritizing finish time in resource, task scheduling, and optimization.

Table 7: Shows the average finishing time with simulation 5 VMs in ms

Num of Tasks	PSO	ACO	DBSCAN	KMPPVM-DRM
1000	699.304	630.304	313.4951	123.24
3000	1926.053	1867.053	911.90058	223.49
5000	3173.058	3129.058	1536.658	499.16
8000	5029.683	4961.683	2413.843	935.47
10000	6321.599	6288.599	3103.799	1230.91

3. Start Time: Tables (9,10) show the average start time values when a resource (such as a virtual machine or container) is allocated to the task and begins executing by various models executing tasks across different counts and VMs. In the PSO model, the average start time increases from 654.315 (1000 tasks) to 6320.839(10000 tasks) for 5 VMs, from 219.44 for

Table 8: Shows the average finishing time with a simulation of 10 VMs in ms

Num of Tasks	PSO	ACO	DBSCAN	KMPPVM-DRM
1000	489.5128	441.2128	219.4466	61.93
3000	1348.237	1306.937	638.3304	117.98
5000	2221.14	2190.34	1075.66	337.58
8000	3520.778	3473.178	1689.69	742.35
10000	4425.119	4402.019	2172.659	1027.27

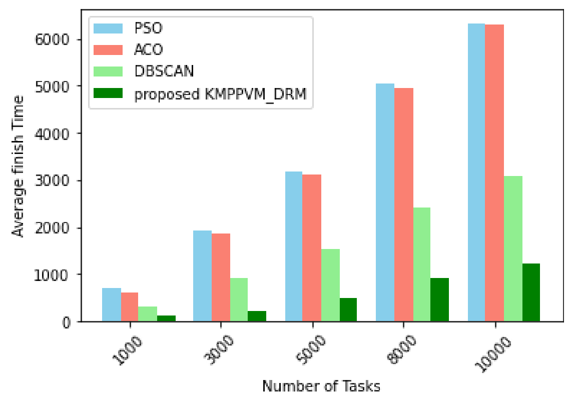


Figure 5: The finish time average for tasks performed by 5 VMs

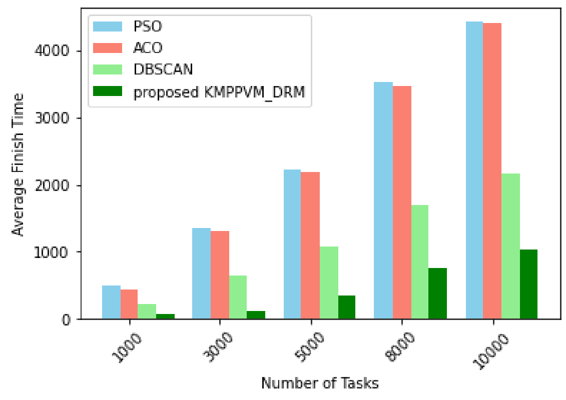


Figure 6: The finish time average for tasks performed by 10 VMs

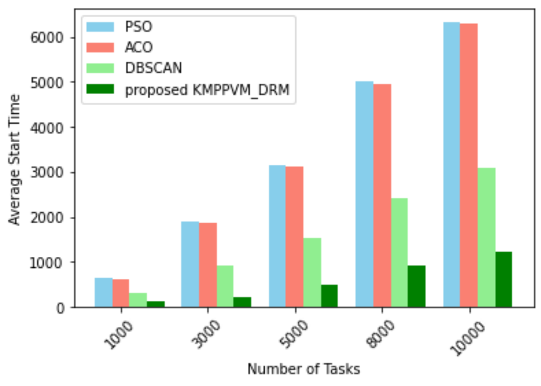
1000 tasks to 2172.65 for 10000 tasks for 10 VMs, indicating a proportional rise in average start time with an increasing task load. The ACO and DBSCAN follow similar trends, with start time average values escalating as the number of tasks grows with (5,10)VMs. Notably, the proposed KMPPVM-DRM model stands out with significantly lower average start time values across all task counts (e.g., 122.87 for 1000 tasks with 5VMs and 61.56 for 1000 tasks with 10 VMs), showcasing its faster task execution response and directly contributing in early RA and efficient scheduling compared to the other models. Figures (7a and 7b) show the average start time of KMPPVM-DRM compared to the other models.

Table 9: Shows the start time average with simulation 5 VMs in ms

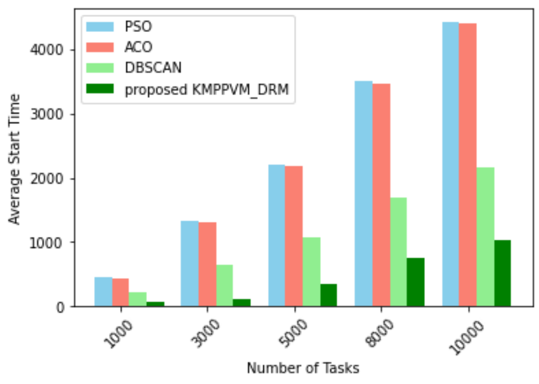
Num of Tasks	PSO	ACO	DBSCAN	KMPPVM-DRM
1000	654.3151	625.31518	308.1501	122.87
3000	1906.083	1862.083	906.5555	222.95
5000	3135.073	3124.073	1531.313	498.72
8000	5002.0138	4956.013	2408.498	935.09
10000	6320.839	6282.839	3098.454	1230.54

Table 10: Shows the start time average with a simulation of 10 VMs in ms

Num of Tasks	PSO	ACO	DBSCAN	KMPPVM-DRM
1000	458.0206	437.72068	215.7051	61.56
3000	1334.258	1303.458	634.5889	117.44
5000	2194.551	2186.851	1071.919	337.13
8000	3501.409	3469.209	1685.949	741.962
10000	4424.587	4397.987	2168.918	1026.9



(a) The start time average for tasks performed by 5 VMs.



(b) The start time average for tasks performed by 10 VMs.

Figure 7: Comparison of the start time average under different VM configurations: (a) 5 VMs, (b) 10 VMs

4.4 Performance analysis

To understand how the proposed algorithm's performance is affected by the amount of available resources, a set of experiments was conducted using two different virtual machine (VM) configurations: one with 5 VMs and the other with 10 VMs, with varying task loads (from 1,000 to 10,000 tasks). This analysis aimed to evaluate the algorithm's adaptability to environments with limited versus larger resources and assess the consistency of the improvement in task execution speed compared to the DBSCAN, ACO, and PSO algorithms. Two analytical techniques were adopted to measure the performance of the proposed algorithm (KMPPVM-DRM): Relative Reduction to estimate the percentage of improvement relative to other algorithms and harmonic mean to find a fair estimate.

4.4.1 Performance analysis with 10 VMs

Table (11) shows the relative reduction in execution time that the suggested algorithm offers compared to others and for various tasks of various sizes. It is noted that the proposed algorithm significantly outperformed all other algorithms, the improvement over the DBSCAN algorithm ranging from 69.90% to 71.52%, compared to the ACO algorithm ranging from 78.64% to 81.95%, and compared to the PSO algorithm ranging from 76.65% to 79.18%. To

Table 11: Shows relative reduction percentages (%) in execution time achieved by KMPPVM-DRM compared to other algorithms, with 10 VMs

Number of Tasks	DBSCAN	ACO	PSO
1000	71.50	81.95	79.18
3000	69.90	78.64	76.65
5000	70.83	80.40	77.37
8000	71.35	80.62	78.32
10000	71.52	80.87	77.96

have an unbiased analysis of the proposed method's performance, especially with significantly varying values, the harmonic mean of the execution time of each method for various task counts was worked out. The harmonic means of the results are summarized in the following table(12).

Table 12: Shows the Harmonic mean of each algorithm's execution time with 10 VMs

Method	Harmonic Mean (ms)
PSO	4102.99
ACO	4672.25
DBSCAN	3071.51
KMPPVM-DRM (Proposed)	**886.72**

The KMPPVM-DRM algorithm achieved the lowest harmonic mean, demonstrating more efficient and consistent performance than the other algorithms, even as the number of tasks increased. This supports the reliability of the relative reduction results in reducing the total execution time and highlights the effectiveness of the proposed method in managing cloud computing resources. Figure (8) shows the harmonic means of KMPPVM-DRM compared to the other models. The results showed that the KMPPVM-DRM algorithm significantly outperformed the other algorithms in reducing both the average finish time and the average start

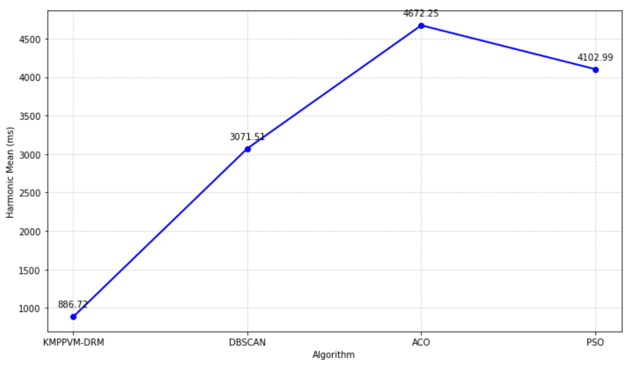


Figure 8: Comparison of Harmonic Means of Algorithms with 10 VMs

time with 10 virtual machines, reflecting the model's efficient use of available resources. Figures (9,10) illustrate the comparative performance regarding average finish and start times, respectively. Furthermore, Tables (13,14) provide a detailed summary of the relative reduction percentages of average finish and start times achieved by the proposed method across different task sizes.

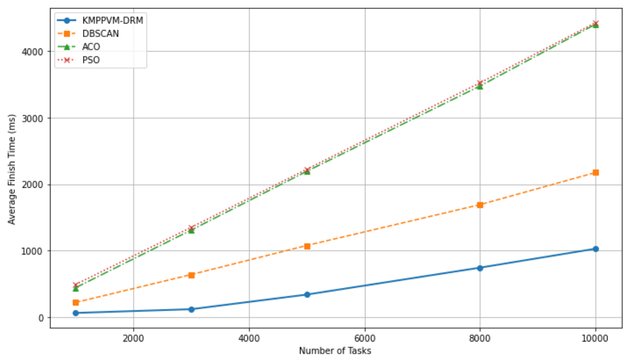


Figure 9: comparative performance for average finish time of Algorithms with 10 VMs

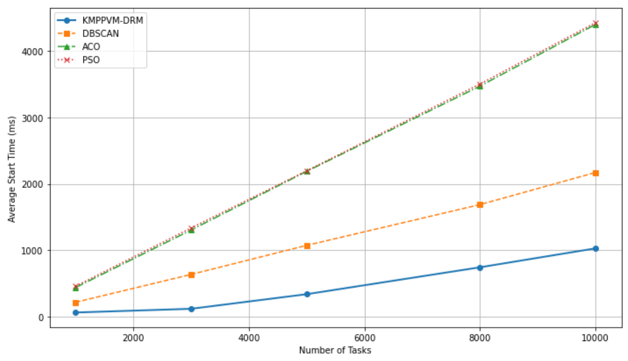


Figure 10: comparative performance for average start time of Algorithms with 10 VMs

Table 13: Shows relative reduction percentages (%) in average finish time achieved by KMPPVM-DRM compared to other algorithms

Number of Tasks	DBSCAN	ACO	PSO
1000	71.77	85.96	87.34
3000	81.51	90.97	91.24
5000	68.61	84.58	84.80
8000	56.06	78.62	78.91
10000	52.71	76.66	76.78

Table 14: Shows relative reduction percentages (%) in average start time achieved by KMPPVM-DRM compared to other algorithms

Number of Tasks	DBSCAN	ACO	PSO
1000	71.46	85.93	86.55
3000	81.49	90.99	91.19
5000	68.54	84.58	84.63
8000	55.99	78.61	78.80
10000	52.65	76.65	76.79

In this table(13), It is noted that the proposed algorithm significantly outperformed all other algorithms in the relative reduction percentages for the average finish time, the improvement over the DBSCAN algorithm ranging from 52.71% to 81.51%, compared to the ACO algorithm ranging from 76.66% to 90.97%, and compared to the PSO algorithm ranging from 76.78% to 91.24%. Table (14) also presents the relative reduction percentages of the average start time, clearly highlighting the superiority of the proposed algorithm over the other algorithms. The improvement over the DBSCAN algorithm ranged from 52.65% to 81.49%, compared to the ACO algorithm, which ranged from 76.65% to 90.99%, and compared to the PSO algorithm, which ranged from 76.79% to 91.19%. Table (15) presents each algorithm's harmonic mean of the average start and finish times. The findings indicate that the proposed KMPPVM-DRM algorithm outperforms the others, recording the lowest harmonic mean values. It demonstrates its enhanced capability to reduce task waiting time and accelerate execution.

Table 15: Shows the harmonic mean of each algorithm's average finish and start times with 10 VMs

Method	Harmonic Mean (Start Time)	Harmonic Mean (Finish Time)
PSO	1282.03	1335.19
ACO	1242.34	1248.98
DBSCAN	609.84	616.88
KMPPVM-DRM	**166.41**	**167.19**

The following figures (11,12) show the harmonic mean of the average times (start and finish times). Figure (11) shows the harmonic mean of the task start average, while Figure (12) highlights the harmonic mean of the task finish average.

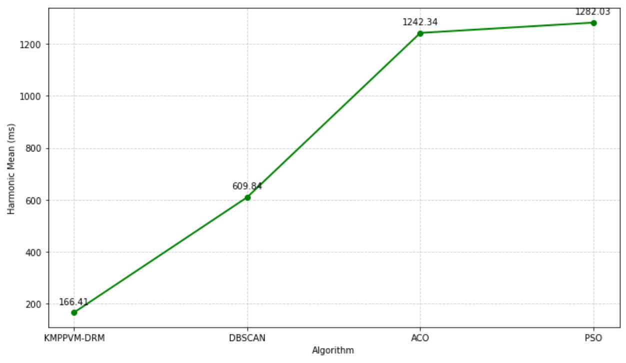


Figure 11: Comparison of harmonic means for average start time of algorithms with 10 VMs

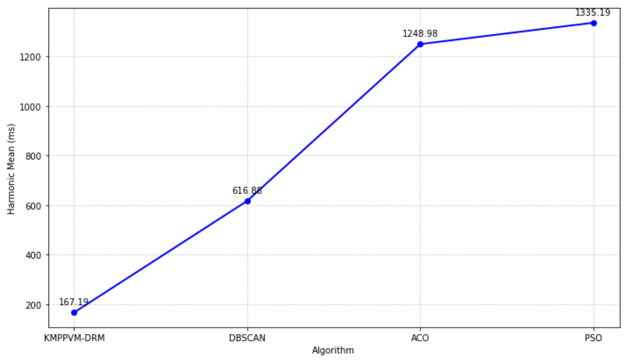


Figure 12: Comparison of harmonic means for average finish time of algorithms with 10 VMs

4.4.2 Performance analysis with 5 VMs

This table(16) shows that the relative reduction percentages are still high, albeit slightly lower than in the case where 10 virtual machines were used. This results from limited resources available at the system's disposal and a lot of pressure on the allocation algorithm to balance it. However, the proposed algorithm did not lose its superiority compared to the other algorithms, demonstrating its efficiency in adapting to the resource-limited environment.

Table 16: Shows relative reduction percentages (%) in average finish and start time achieved by KMPPVM-DRM compared to other algorithms, with 5 VMs

Method	Number of Tasks	Avg. Start Time (ms)	Avg. Finish Time (ms)
PSO	1000	81.22	82.37
	3000	88.30	88.39
	5000	84.09	84.26
	8000	81.30	81.40
	10000	80.53	80.52
ACO	1000	80.35	80.44
	3000	88.01	88.02
	5000	84.03	84.04
	8000	81.13	81.14
	10000	80.41	80.42
DBSCAN	1000	60.12	60.68
	3000	75.40	75.49
	5000	67.43	67.51
	8000	61.17	61.24
	10000	60.28	60.34

Table (17) presents each algorithm's harmonic mean of the average start and finish times. The findings indicate that the proposed KMPPVM-DRM algorithm outperforms the others, recording the lowest harmonic mean values with 5 VMs. It demonstrates its enhanced capability to reduce task waiting time and accelerate execution.

Table 17: Shows the harmonic mean of each algorithm's average finish and start times with 5 VMs

Method	Harmonic Mean (Start Time)	Harmonic Mean (Finish Time)
PSO	1831.47	1907.41
ACO	1774.77	1784.26
DBSCAN	871.20	881.26
KMPPVM-DRM (Proposed)	**302.82**	**303.52**

Compared to the 10-VM use case, the harmonic mean values for the average start and end times increased for all algorithms as resources were reduced, increasing the pressure on the hardware and leading to processing delays. However, the proposed algorithm (KMPPVM-DRM) maintained the best relative performance with the lowest harmonic mean value, confirming its high capacity and flexibility in resource-constrained cloud computing environments. The following figures (13,14) show the harmonic mean of the average times (start and finish times) with 5VMs. Figure (13) shows the harmonic mean of the task start average, while Figure (14) highlights the harmonic mean of the task finish average. The results revealed that KMPPVM-DRM produces vastly improved performance compared to other algorithms in the three metrics of the total execution time and average finish and start times while significantly reducing the execution time. The data also show that such improvements significantly increase with the increasing number of virtual machines, demonstrating the model's effective scalability.

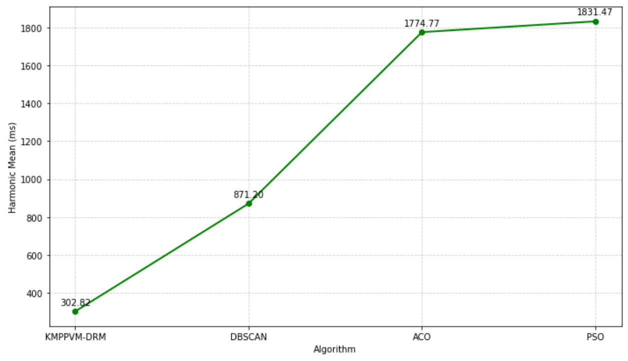


Figure 13: Comparison of harmonic means for average start time of algorithms with 5 VMs

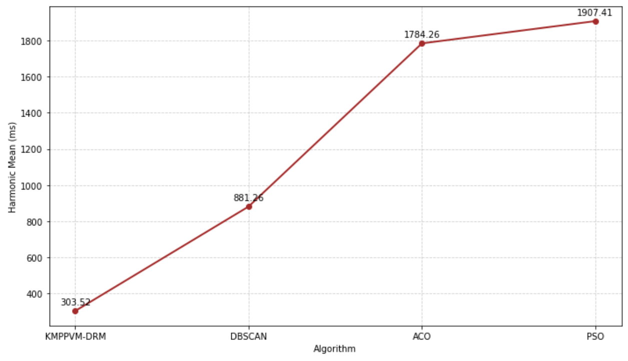


Figure 14: Comparison of harmonic means for average finish time of algorithms with 5 VMs

4.5 Statistical analysis using paired T-test (with Multiple VMs)

In this study section, a paired t-test was used to test for the statistical significance of the performance differences between the proposed algorithm (KMPPVM-DRM) and the other algorithms (DBSCAN, ACO, PSO) in two separate scenarios. The first has 5 virtual machines, and the second has 10. This analysis determines whether the observed differences across various task loads reflect genuine improvements or are simply due to random variation. The t-test results showed that the difference was significant at 0.05 level in both scenarios while supporting the effectiveness and flexibility of the suggested algorithm under a resource-limited environment. Table (18) presents the results of the conducted statistical analysis with a paired t-test for performance comparisons between the proposed KMPPVM-DRM algorithm and other algorithms based on three major performance indicators (execution time, average start, and finish time) in a computing environment comprising 10 virtual machines (VMs).

Table 18: Shows paired T-test results (t-statistic, p-value) for KMPPVM-DRM vs other algorithms on 10 VMs

Metric	Comparison	t-statistic	p-value
Execution Time	KMPPVM-DRM vs DBSCAN	-3.28	0.0303
	KMPPVM-DRM vs ACO	-3.27	0.0307
	KMPPVM-DRM vs PSO	-3.30	0.0297
Average Start Time	KMPPVM-DRM vs DBSCAN	-4.07	0.0152
	KMPPVM-DRM vs ACO	-3.56	0.0234
	KMPPVM-DRM vs PSO	-3.60	0.0227
Average Finish Time	KMPPVM-DRM vs DBSCAN	-4.09	0.0149
	KMPPVM-DRM vs ACO	-3.57	0.0233
	KMPPVM-DRM vs PSO	-3.66	0.0215

The results in Table (18) revealed that all p-values were smaller than 0.05, thus indicating the statistical significance of the successes of KMPPVM-DRM. This indicates the superiority and stability of the algorithm's performance on different metrics. Additionally, all comparisons' t-statistic values were negative, indicating that KMPPVM-DRM consistently outperformed the other algorithms regarding reducing time metrics. The figures(15,16,17) below also show the p-values of paired t-test results between KMPPVM-DRM and the other algorithms across three-time metrics: execution time, average finish time, and average start time. All p-values shown in the figures were less than the statistical significance level of 0.05, confirming the presence of statistical significance and superiority in favour of the proposed algorithm.

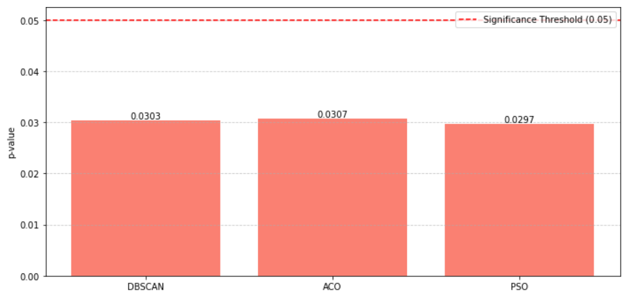


Figure 15: p-values of paired t-test results between KMPPVM-DRM and other algorithms for execution time with 10 VMs

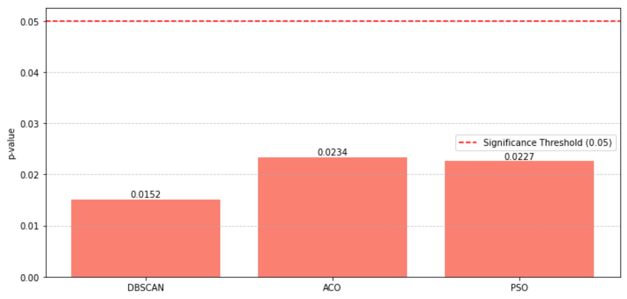


Figure 16: p-values of paired t-test results between KMPPVM-DRM and other algorithms for Average Start time with 10 VMs

In order to strengthen the assessment of the efficiency of the proposed KMPPVM-DRM algorithm, the paired t-tests were performed to compare the algorithm to other algorithms in terms of performance metrics (execution time, average finish time, and average start time) at resource-constrained settings (5VMs). The results of the paired t-tests in Table (19) indicate that the p-values have a significance level of less than 0.05, confirming that the observed improvements are statistically significant. On the other hand, negative t values show that KMPPVM-DRM had better performance metrics, demonstrating its efficiency and adaptability even in resource-constrained environments.

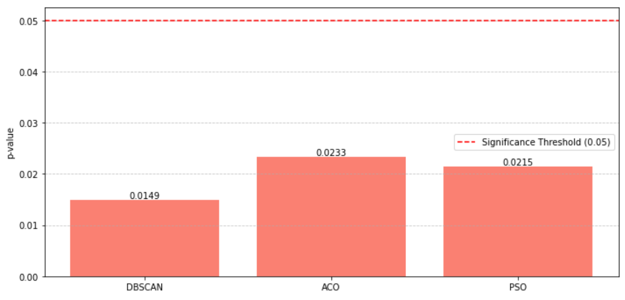


Figure 17: p-values of paired t-test results between KMPPVM-DRM and other algorithms for Average Finish time with 10 VMs

Table 19: Shows paired T-test results (t-statistic, p-value) for KMPPVM-DRM vs other algorithms on 5 VMs

Metric	Comparison	t-statistic	p-value
Execution Time	KMPPVM-DRM vs DBSCAN	-3.32	0.0292
	KMPPVM-DRM vs ACO	-3.25	0.0312
	KMPPVM-DRM vs PSO	-3.29	0.0302
Average Start Time	KMPPVM-DRM vs DBSCAN	-3.56	0.0235
	KMPPVM-DRM vs ACO	-3.40	0.0271
	KMPPVM-DRM vs PSO	-3.43	0.0263
Average Finish Time	KMPPVM-DRM vs DBSCAN	-3.58	0.0231
	KMPPVM-DRM vs ACO	-3.41	0.0270
	KMPPVM-DRM vs PSO	-3.49	0.0250

The figures(18,19) below also show the paired t-test results between KMPPVM-DRM and the other algorithms, and All p-values shown in the figures were less than the statistical significance level of 0.05, confirming the presence of statistical significance and superiority in favour of the proposed algorithm even with the environment resource-constrained (5VMs).

4.6 Discussion and interpretation

Compared to DBSCAN, ACO, and PSO, the KMPPVM-DRM algorithm ran faster, reduced execution time, and improved the average start and finish time and the distribution of jobs among processors. The boost in performance comes from using KMeans++ to cluster VMs, which helps choose much better centres and facilitates faster convergence. Unlike DBSCAN, which struggles with irregular cloud workloads and requires sensitive parameter tuning, ACO/PSO is time-consuming. KMPPVM-DRM clustering of VMs into groups (high, medium, low) and distributes tasks to the most appropriate VM in each group based on demand. The

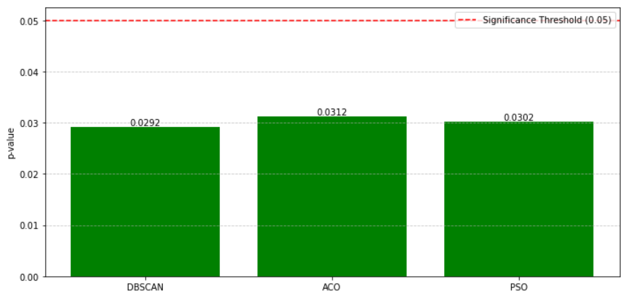
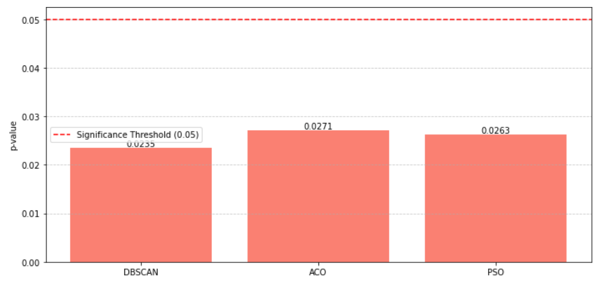
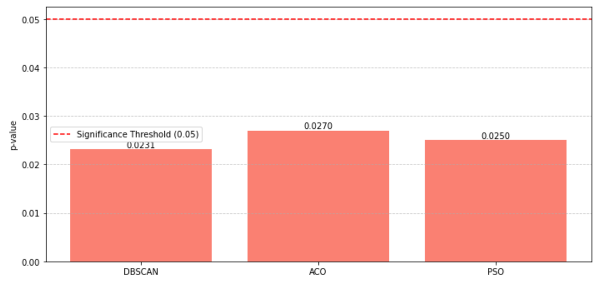


Figure 18: p-values of paired t-test results between KMPPVM-DRM and other algorithms for execution time with 5 VMs



(a) p-values of paired t-test to (average start time)



(b) p-values of paired t-test to (Average Finish time).

Figure 19: p-values of paired t-test results between KMPPVM-DRM and other algorithms: (a) average start time, (b) average finish time with 5 VMs

method uses KMeans++ to make strong VM groups, which offers better initialization, computational efficiency, and stability under real-time VM load variations and dynamic resource management (DRM) to manage the intelligent allocation of tasks. VMs are sorted by their capabilities, and specific tasks are assigned according to these groupings. Meanwhile, VM weights change in real-time with changes in the system. This enables improved scheduling decisions. Because KMPPVM-DRM has a smaller search area and less to worry about than ACO and PSO, it is more efficient regarding computational complexity. Nevertheless, handling massive cloud deployments and continually updating the weights can be difficult. However, these issues might be handled by parallel clustering techniques or leveraging distributed computing frameworks in the future. Overhead Consideration: While the current version does not explicitly measure the computation time of KMeans++, scheduler response time, or system latency, these aspects are acknowledged as potential overhead sources. Future experiments will include benchmarks for runtime overhead, especially under large-scale conditions, to quantify the impact of clustering and dynamic weight updates on overall system responsiveness.

5 Conclusion and future direction

This paper develops a system model that optimizes dynamic resource scheduling procedures through efficient cloud resource allocation(RA) with support for different cloud infrastructure deployments. Tasks are classified based on their length property, assuming a scenario with three different task length levels: high, medium, and low. For the allocation, a dynamic clustering mechanism(KMeans++) is used based on the current load weight of virtual machines (VMs) in real-time. In a cluster of machines, a ULC is used to identify the most suitable virtual machine from each cluster to be assigned to execute the task by finding the machine with the most significant weight of the available resources in the cluster. As shown in the results section, the proposed model outperforms the existing methods in (DBSCAN, ACO, and PSO). The performance evaluation relies on execution time measurements alongside start time and finish time averages. The result shows that the proposed method performs better than the existing algorithms in (DBSCAN, ACO, and PSO) for both execution time and start time and finish time averages for different numbers of tasks ranging from (1000 to 10000) with (5 and 10) virtual machines. The results validate the efficiency and adaptability of the proposed model for heterogeneous cloud infrastructures. Using 10 virtual machines, the proposed model had the best harmonic execution time of 886.72, outperforming PSO at 4102.99, ACO at 4672.25 and DBSCAN at 3071.51. Similarly, the harmonic means of average start and finish times were significantly lower at 166.41 and 167.19, respectively, whereas other methods exhibited values exceeding 600 and 1200. In terms of relative reduction percentages as an estimate of performance improvement, measured by reducing the execution time, was from 69.90% to 71.52% for the proposed model compared to DBSCAN, meaning the model is dependable no matter how many tasks it needs to deal with (up to 10000). Compared to DBSCAN, the reduction in finish time average was between 52.71% and 81.51%, and the start time average was from 52.65% to 81.49%. Moreover, even under limited resource conditions (with only 5 virtual machines), the proposed approach demonstrated competitive and stable performance compared to the baseline algorithms, highlighting its robustness and adaptability.

The proposed model will be extended in future work to support larger-scale environments with increased numbers of 100+ VMs, variable task inter-arrival rates, diverse task types, and VM failure scenarios. In addition, examining the use of additional machine learning algorithms (may consider integrating predictive analytics, such as Cloud Linear Regression (CLR) models as introduced in [32], [33], and [34], to enhance the accuracy of resource demand forecasting) to optimize RA and load further balancing in cloud data centres. Finally, sensitivity analysis will be conducted on the CPU and RAM weight parameters (α , β) to identify optimal configurations for varying workload types and deployment scenarios.

References

[1] H. S. Malallah, R. Qashi, L. M. Abdulrahman, M. A. Omer, and A. A. Yazdeen, "Performance analysis of enterprise cloud computing: a review," *Journal of Applied Science and Technology Trends*, vol. 4, no. 01, pp. 01–12, 2023. https://doi.org/10.38094/jastt401139.

[2] S. Paul and M. Adhikari, "Dynamic load balancing strategy based on resource classification technique in iaas cloud," in *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 2059–2065, IEEE, 2018. https://doi.org/10.1109/icacci.2018.8554440.

[3] S. S. Gill and R. Buyya, "Resource provisioning based scheduling framework for execution of heterogeneous and clustered workloads in clouds: from fundamental to autonomic offering," *Journal of Grid Computing*, vol. 17, pp. 385–417, 2019. https://doi.org/10.1007/s10723-017-9424-0.

[4] A. Shahidinejad, M. Ghobaei-Arani, and M. Masdari, "Resource provisioning using workload clustering in cloud computing environment: a hybrid approach," *Cluster Computing*, vol. 24, no. 1, pp. 319–342, 2021. https://doi.org/10.1007/s10586-020-03107-0.

[5] O. H. Sultan and T. Khaleel, "Challenges of load balancing techniques in cloud environment: A review," *Al-Rafidain Engineering Journal*, vol. 27, no. 2, pp. 227–235, 2022. https://doi.org/10.33899/rengj.2022.134056.1179.

[6] R. Tasneem and M. Jabbar, "An insight into load balancing in cloud computing," in *International Conference on Wireless Communications, Networking and Applications*, pp. 1125–1140, Springer, 2021. https://doi.org/10.1007/978-981-19-2456-9_113.

[7] N. K. Mishra and N. Mishra, "Load balancing techniques: need, objectives and major challenges in cloud computing-a systematic review," *International Journal of Computer Applications*, vol. 131, no. 18, pp. 0975–8887, 2015. https://doi.org/10.5120/ijca2015907523.

[8] Y. Lohumi, D. Gangodkar, P. Srivastava, M. Z. Khan, A. Alahmadi, and A. H. Alahmadi, "Load balancing in cloud environment: A state-of-the-art review," *Ieee Access*, vol. 11, pp. 134517–134530, 2023. https://doi.org/10.1109/access.2023.3337146.

[9] A. M. Manasrah and H. Ba Ali, "Workflow scheduling using hybrid ga-pso algorithm in cloud computing," *Wireless Communications and Mobile Computing*, vol. 2018, no. 1, p. 1934784, 2018. https://doi.org/10.1155/2018/1934784.

[10] A. Senthil Kumar and M. Venkatesan, "Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment," *Wireless Personal Communications*, vol. 107, pp. 1835–1848, 2019. https://doi.org/10.1007/s11277-019-06360-8.

[11] U. K. Jena, P. K. Das, and M. R. Kabat, "Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 2332–2342, 2022. https://doi.org/10.1016/j.jksuci.2020.01.012.

[12] F. Davami, S. Adabi, A. Rezaee, and A. M. Rahmani, "Distributed scheduling method for multiple workflows with parallelism prediction and dag prioritizing for time constrained cloud applications," *Computer networks*, vol. 201, p. 108560, 2021. https://doi.org/10.1016/j.comnet.2021.108560.

[13] P. Paknejad, R. Khorsand, and M. Ramezanpour, "Chaotic improved picea-g-based multi-objective optimization for workflow scheduling in cloud environment," *Future Generation Computer Systems*, vol. 117, pp. 12–28, 2021. https://doi.org/10.1016/j.future.2020.11.002.

[14] Z. Mahmoudi, E. Darbanian, and M. Nickray, "Computational resource allocation in iot fog computing using teaching–learning-based optimization algorithm," *Journal of Soft Computing and Information Technology*, vol. 10, no. 3, pp. 73–85, 2021.

[15] A. Daghayeghi and M. Nickray, "Nsgaii-based task scheduling model for smart city applications in cloud-fog environment," *Journal of Soft Computing and Information Technology*, vol. 11, no. 3, pp. 64–82, 2022.

[16] S. A. Omranian and M. Goudarzi, "Greedy algorithm for dynamic allocation of intelligent services in vehicular edge computing," *Cluster Computing*, vol. 28, no. 1, pp. 1–20, 2025. https://doi.org/10.1007/s10586-024-04817-5.

[17] P. R. Kaveri and P. Lahande, "Reinforcement learning to improve resource scheduling and load balancing in cloud computing," *SN Computer Science*, vol. 4,

no. 2, p. 188, 2023. https://doi.org/10.1007/s42979-022-01609-9.

[18] M. R. Belgaum, S. Musa, M. M. Alam, and M. M. Su'ud, "A systematic review of load balancing techniques in software-defined networking," *Ieee Access*, vol. 8, pp. 98612–98636, 2020. https://doi.org/10.1109/access.2020.2995849.

[19] J. Shankar, I. Hussain, S. Zafar, I. R. Khan, and A. Khalique, "Effective resource allocation and load balancing in green cloud computing," in *International Conference on ICT for Digital, Smart, and Sustainable Development*, pp. 423–439, Springer, 2024. https://doi.org/10.1007/978-981-97-7831-7_26.

[20] Q. Shang, "A dynamic resource allocation algorithm in cloud computing based on workflow and resource clustering," *Journal of Internet Technology*, vol. 22, no. 2, pp. 403–411, 2021.

[21] S. El Motaki, A. Yahyaouy, H. Gualous, and J. Sabor, "A new weighted fuzzy c-means clustering for workload monitoring in cloud datacenter platforms," *Cluster Computing*, vol. 24, no. 4, pp. 3367–3379, 2021. https://doi.org/10.1007/s10586-021-03331-2.

[22] J. P. B. Mapetu, L. Kong, and Z. Chen, "A dynamic vm consolidation approach based on load balancing using pearson correlation in cloud computing," *The Journal of Supercomputing*, vol. 77, no. 6, pp. 5840–5881, 2021. https://doi.org/10.1007/s11227-020-03494-6.

[23] L. M. Al Qassem, T. Stouraitis, E. Damiani, and I. A. M. Elfadel, "Proactive random-forest autoscaler for microservice resource allocation," *IEEE Access*, vol. 11, pp. 2570–2585, 2023. https://doi.org/10.1109/access.2023.3234021.

[24] G. Senthilkumar, K. Tamilarasi, N. Velmurugan, and J. Periasamy, "Resource allocation in cloud computing," *Journal of Advances in Information Technology*, vol. 14, no. 5, pp. 1063–1072, 2023. https://doi.org/10.12720/jait.14.5.1063-1072.

[25] M. S. Al Reshan, D. Syed, N. Islam, A. Shaikh, M. Hamdi, M. A. Elmagzoub, G. Muhammad, and K. H. Talpur, "A fast converging and globally optimized approach for load balancing in cloud computing," *IEEE Access*, vol. 11, pp. 11390–11404, 2023. https://doi.org/10.1109/access.2023.3241279.

[26] S. Singh, P. Singh, and S. Tanwar, "Energy aware resource allocation via ms-slno in cloud data center," *Multimedia Tools and Applications*, vol. 82, no. 29, pp. 45541–45563, 2023. https://doi.org/10.1007/s11042-023-15521-8.

[27] D. Kavitha, M. Priyadharshini, R. Anitha, S. Suma, V. Prema, and A. Vidhya, "Adaptive dbn using hybrid bayesian lichtenberg optimization for intelligent task allocation," *Neural Processing Letters*, vol. 55, no. 4, pp. 4907–4931, 2023. https://doi.org/10.1007/s11063-022-11071-6.

[28] S. D. S. Mustapha and P. Gupta, "Dbscan inspired task scheduling algorithm for cloud infrastructure," *Internet of Things and Cyber-Physical Systems*, vol. 4, pp. 32–39, 2024. https://doi.org/10.1016/j.iotcps.2023.07.001.

[29] I. Jaya, Y. Li, and W. Cai, "Deep reinforcement learning based resource allocation in edge-cloud gaming," *Multimedia Tools and Applications*, vol. 83, no. 26, pp. 67903–67926, 2024. https://doi.org/10.1007/s11042-024-18337-2.

[30] H. Lin, G. Liu, W. Lin, X. Wang, and X. Wang, "A novel virtual machine consolidation algorithm with server power mode management for energy-efficient cloud data centers," *Cluster Computing*, vol. 27, no. 8, pp. 11709–11725, 2024. https://doi.org/10.1007/s10586-024-04555-8.

[31] A. R. Khan, "Dynamic load balancing in cloud computing: optimized rl-based clustering with multi-objective optimized task scheduling," *Processes*, vol. 12, no. 3, p. 519, 2024. https://doi.org/10.3390/pr12030519.

[32] O. K. J. Mohammad, M. E. Seno, and B. N. Dhannoon, "Detailed cloud linear regression services in cloud computing environment," *Informatica*, vol. 48, no. 12, 2024. https://doi.org/10.31449/inf.v48i12.6771.

[33] J. Ma, C. Zhu, Y. Fu, H. Zhang, and W. Xiong, "Dynamic routing via reinforcement learning for network traffic optimization," *Informatica*, vol. 49, no. 8, 2025. https://doi.org/10.31449/inf.v49i8.7126.

[34] C. Vijaya and P. Srinivasan, "Hybrid fuzzy metaheuristic technique for efficient vm selection and migration in cloud data centers," *Informatica*, vol. 48, no. 20, 2024. https://doi.org/10.31449/inf.v48i20.6549.

[35] M. A. Alworafi, A. Dhari, A. A. Al-Hashmi, A. B. Darem, *et al.*, "An improved sjf scheduling algorithm in cloud computing environment," in *2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT)*, pp. 208–212, IEEE, 2016. https://doi.org/10.1109/iceeccot.2016.7955216.

[36] P. Gupta and S. P. Ghrera, "Trust and deadline aware scheduling algorithm for cloud infrastructure using ant colony optimization," in *2016*

International conference on innovation and challenges in cyber security (ICICCS-INBUSH), pp. 187–191, IEEE, 2016. https://doi.org/10.1109/iciccs.2016.7542337.

[37] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning pso-based deadline constrained task scheduling for hybrid iaas cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, 2013. https://doi.org/10.1109/tase.2013.2272758.