Hindawi
Applied Bionics and Biomechanics
Volume 2023, Article ID 4350615, 13 pages
https://doi.org/10.1155/2023/4350615

Research Article

Multiobjective Prioritized Workflow Scheduling in Cloud Computing Using Cuckoo Search Algorithm

Babuli Sahu,[1] Sangram Keshari Swain,[1] Sudheer Mangalampalli ,[2] and Satyasis Mishra [3]

[1]*Department of CSE, Centurion University of Technology and Management, Bhubaneswar, Odisha, India*
[2]*School of Computer Science and Engineering, VIT-AP University, Amaravati, Andhra Pradesh, India*
[3]*Adama Science and Technology University, Adama, Ethiopia*

Correspondence should be addressed to Satyasis Mishra; satyasismishra@gmail.com

Received 6 February 2023; Revised 17 June 2023; Accepted 21 June 2023; Published 7 July 2023

Academic Editor: Raimondo Penta

Effective workflow scheduling in cloud computing is still a challenging problem as incoming workflows to cloud console having variable task processing capacities and dependencies as they will arise from various heterogeneous resources. Ineffective scheduling of workflows to virtual resources in cloud environment leads to violations in service level agreements and high energy consumption, which impacts the quality of service of cloud provider. Many existing authors developed workflow scheduling algorithms addressing operational costs and makespan, but still, there is a provision to improve the scheduling process in cloud paradigm as it is an nondeterministic polynomial-hard problem. Therefore, in this research, a task-prioritized multiobjective workflow scheduling algorithm was developed by using cuckoo search algorithm to precisely map incoming workflows onto corresponding virtual resources. Extensive simulations were carried out on workflowsim using randomly generated workflows from simulator. For evaluating the efficacy of our proposed approach, we compared our proposed scheduling algorithm with existing approaches, i.e., Max–Min, first come first serve, minimum completion time, Min–Min, resource allocation security with efficient task scheduling in cloud computing-hybrid machine learning, and Round Robin. Our proposed approach is outperformed by minimizing energy consumption by 15% and reducing service level agreement violations by 22%.

1. Introduction

Cloud computing is an emerging area that has changed the requirements of computing systems from local servers to cloud-based systems by using a huge number of heterogeneous distributed resources. A client required high-speed internet access to avail of all the facilities. Cloud systems are classified into (a) public clouds, (b) private clouds, (c) hybrid clouds, and (d) community clouds as per the users' requirements. A private cloud is deployed to provide or keep highly secure data, whereas a public cloud is open for the public to get the service on a pay-as-you-go basis. A hybrid cloud system is partially private and partially public, which means secured information will be stored in private and the rest in public. A community cloud system is meant for a group of people working in the same community.

Depending on the services available in the cloud [1], they are classified as (a) Platform as a Service (PaaS), (b) Software as a Service (SaaS), and (c) Infrastructure as a Service (IaaS). IaaS provides clients with different resources like storage, networks, and processing capacities. PaaS gives different platforms, like Windows, UNIX, Linux, etc., and SaaS provides enterprise software to clients. As there are many tasks to be performed in a cloud system, and they are mostly dependent on each other, either functionally or based on the shared resources they will use, called workflow, it is a challenge for the service providers. The workflow schedule is an nondeterministic polynomial (NP)-hard optimization problem. A well-developed workflow algorithm is essential, as numerous tasks can be performed without compromising SLA violations and decreasing energy consumption. Budget, energy, security, deadline constraints, and fault tolerance are the essential parameters for workflow scheduling. In the execution of a task in a cloud environment, it is primarily two-phase. The phases are provisioning different resources and mapping the resources to the task. Resource provisioning is mostly

performed using dynamic scheduling and static scheduling. In static scheduling, the required configurations are well known, so they are challenging to implement as per the requirements. However, in dynamic scheduling, the scalability of resources is used to face uncertain challenges. The scalability of the resource configuration is a challenge for researchers. Various algorithms are developed to manage the scalability, looking into the significant parameters. The parameters of the research are classified into three categories: single-objective, bi-objective, and multiobjective based on the user's needs [2].

A distributed consolidation is more scalable than a centralized cloud system, but the message passing among the physical machines (PM) may lead to network and bandwidth issues in the cloud system. A centralized dynamic consolidation is more energy efficient in a cloud system. Dynamic consolidation is categorized into two types: (a) nonproductive, in which the algorithm considers the total sum of current resource utilization but does not predict future resource utilization. On the other hand, (b) predictive dynamic consolidation predicts future workloads based on predictive techniques. Workflow-based virtual machine (VM) placement is an essential task in cloud computing because it optimizes workloads and reduces costs and SLA violations. As per the research papers published by the researchers, the problems in VM placement can be classified as "fresh placement," "initial placement," or "static placement." In this type of placement, there will be continuous requests for deploying VMs on the empty hosts, along with removal requests for the VMs. In this type of situation, replying to the request system in a short amount of time is highly important. In the second case, consolidated placement or dynamic consolidation, some of the tasks based on workloads will be finished, and some new workloads will be requested. This is an ongoing process. So optimization is needed to solve the energy consumption and SLA violation issues [3]. In the scheduling process, the system will map the execution of interdependent tasks across the distributed systems. The scheduler will allocate appropriate and necessary resources to the workflow-based system so that the system will minimize the execution time. As specific solutions to the problem and solutions to it could be more practical due to the overhead of scheduling generation, which is very high. There are no known algorithms to generate the most optimal solutions within polynomial-time scheduling, so distributed resources are considered NP-hard problems [4]. Different types of meta-heuristic algorithms are used in cloud systems, such as genetic algorithm (GA), particle swarm optimization (PSO), and artificial bee colony algorithms are used to solve the scheduling of the workflow tasks in the cloud environments [5]. Most researchers have focused on parameters like makespan, reliability, computational cost, and processing costs. They have yet to, however, concentrate on other matrices such as memory utilization, electricity cost, and network utilization [6]. Many of the existing authors proposed various scheduling algorithms and addressed various parameters, but workflow scheduling in cloud computing is an NP-hard problem, and still, there is a chance to improve the scheduling process by fine-tuning parameters that impact the scheduling process. Therefore, in this manuscript, we used crow search algorithm to tackle the scheduling process by considering the priorities of tasks and their dependencies to accurately map these workflows onto appropriate virtual resources.

The main contributions and limitations of this manuscript are mentioned below:

(1) A multiobjective workflow scheduling model developed by using crow search algorithm.
(2) Task priorities are calculated based on the length of task and the processing capacity of VM.
(3) Extensive simulations are conducted on workflowsim by generating random workloads.
(4) Evaluated our proposed approach with baseline approaches to identify the efficacy of our proposed approach with respect to the metrics concerned, i.e., SLA violation and energy consumption.
(5) The main limitation of our proposed research is our proposed scheduler is not able to predict upcoming workloads which may come to cloud console. Therefore, in future, we need to employ a machine-learning model in scheduler to improve the scheduling process.

Our paper is divided into several sections, including related work in Section 2, various performance metrics used in Section 3, the proposed algorithm in Section 4, experimental setup in Section 5, research outcome and discussion in Section 6, and conclusion and future work in Section 7.

2. Related Works

In the paper, the researchers focused on two consolidation-based energy-efficient techniques. They named the algorithms maximum capacity best fit decreasing and minimum power best fit decreasing. They improve on the existing algorithm by incorporating the enhanced-conscious task consolidation method and the maximum utilization (MaxUtil) technique to reduce energy consumption and SLA violations. The results of their experiments show that the proposed algorithm is better in terms of energy-saving, SLA violation, and VM migrations. The algorithm selects the best server based on the energy consumption and capacity of the CPU available on the server.

They used the upper threshold value by keeping free resources for future usage, managing the ever-changing demands, and reducing SLA violations [7]. The researchers proposed an online workflow scheduling algorithm to manage the multicloud environments' scientific workflow. The proposed algorithm is based on adaptive resource allocation and consolidation, named OWS-A2C. The scientific workflow gets executed in the local perspective and then, according to the requirements of multiple softwares, the instances are allocated according to the process. They compare the proposed algorithm results with those of GAINM, PHGS, and PHA2CI and show that their proposed algorithm's average utilization of the resource is less than the others and consumes less energy [8]. The researchers focused on two major parameters: low cost and makespan for workflow scheduling. They proposed an immune particle swarm

optimization algorithm named immune mutation particle swarm optimization to improve the speed and quality of the optimization. The proposed method is based on encoded mode, affinity calculation, antibody concentration, incentive function, antibody cloning, crossing, and mutation. As per the experimental result shown by the researchers and the outcome of the proposed algorithm, it is clear that the algorithm is better in terms of makespan and cost of computing. They compared their algorithm with PSO and Iass cloud infrastructure for deadline constraint tasks for cloud provider to show that the proposed algorithm is more efficient in terms of saving cost and decreasing the makespan [9]. The researchers proposed a budget-constrained workflow-based scheduling algorithm named task-driven scheduling algorithm (TDSA) to optimize the makespan time in the workflow scheduling. They used two novel mechanisms: (a) a dynamic sub-budget allocation mechanism for the task and (b) task scheduling based on task duplication in the scheduling mechanism. They used the sub-budget allocation method to recover the unused budget for scheduling. In the supplication-based task scheduling mechanism, they explode idle slots in the resources to duplicate the tasks selectively. The experiment results show that it optimizes the makespan up to 17.5% and resource utilization up to 31.6%. The researchers compared their algorithm with greedy resource provisioning and modified heterogeneous earliest finish time (GRP-HEFT), fair budget constrained workflow scheduling approach for heterogeneous clouds, task-dependent scheduling algorithm for N databases, and earliest finish time multiple efficient resource and found that the proposed algorithm saves more budgets and reduces the makespan [10].

The algorithm proposed by the researchers is based on reducing energy consumption in the cloud using critical task remapping (RMREC), which is mainly considered execution time or cost under a budget constraint. They decomposed the algorithm into two different phases: (a) reduction of energy consumption and (b) critical task remapping. To achieve the task, a preliminary mapping between the tasks and VMs is based on the lowest energy consumption. In the second phase of the process, the critical tasks are remapped to the VMs with lower share costs and energy consumption. The researcher compares their proposed algorithm with minimizing energy consumption algorithm with budget constraint and minimizing energy consumption with budget load balancing. The result shows that the proposed algorithm outperforms cost and energy minimization [11]. The researchers focused on efficient scheduling based on task classification and threshold. They have divided the algorithms into phases in which the first phase is to process the workflow tasks to avoid the bottlenecks of dependencies and long execution times. In the second, they used PSO to select the best schedule. The researchers compared their proposed algorithm with the GA, PSO, energy efficient load balancing algorithm with workflow scheduling, and firefly and improved particle swarm optimization and found that the algorithm proposed by the researchers is much better in energy saving, makespan, and load balancing. Further research on VM migration and adaptive threshold requirements needs to be analyzed further to get more energy consumption [12]. In the article, the researchers used a clustering approach to minimize SLA violations. They focused on SLA violation prediction and workload prediction violation using Naive Bayer's classification approach. They compare the proposed algorithm results with those of active VMs without prediction and with the autoregressive-integrated moving average approach for VM performance. From the experimental results, it is clear that the SLA violation in the proposed algorithm is less than the other methods, and it performed better than the other state-of-the-art techniques [13]. The researcher proposes novel resource provisioning for different tasks and workflow scheduling in the paper. They named the proposed algorithms as novel resource provisioning and workflow scheduling, GRP-HEFT. They tried to minimize the amount of time in the cloud under a certain budget. They compare the algorithm with other state-of-the-art workflows like multiobjective ant colony scheduling, PSO, and GA. When they compared the makespan to the time complexity, they discovered that the proposed algorithm outperforms others by more than 13%. The algorithm's time complexity is O (mN), and in the worst case, it is O (mN2). Another advantage of the proposed algorithm is that it is deterministic, meaning that every run with the same input derives the same makespan. Further research work is required to leverage Spot instances [14]. The researchers focused on a budget-constrained workflow-based scheduling algorithm to decrease the makespan. As the billing model in most cloud computing systems is based on hours, and it is a day-to-day challenge for workflow scheduling, it results in a higher makespan and also infeasible solutions. The proposed algorithm is based on an hourly-based billing cycle. Besides that, as data constraints apply to workflow tasks in the cloud, there is a possibility of idle slots in the cloud resources. A few works could be utilized for these idle slots so that duplicate tasks are predecessors to reduce completion time. This will minimize the workflow's makespan while ensuring budget constraints. The proposed TDSA model is based on two important factors: the dynamic sub-budgeting algorithm and the first one is mostly responsible for recovering the unused budget and redistributing it, and the second one is based on a task duplication scheduling mechanism to make use of the idle system slots on the resources to duplicate the task selectively. The result of the algorithm shows that the TDSA algorithm decreases the makespan by 17.4% and resource utilization by 31.6% [15]. As workflow management and scheduling in cloud computing are very urgent, the researcher focused on smart cloud management to increase performance. They compared the proposed algorithm with GA algorithms. They proposed the modified ant colony optimization (MOACO) and cheetah chase algorithm (CCA) algorithms to increase CPU resource utilization and minimize response time. The researcher used a dynamic resource prediction model to increase the performance [16]. The researchers developed novel algorithms to minimize the makespan using two mechanisms: dynamic sub-budget allocation in which the unused budgeting is scheduled, and task duplication-based scheduling, in which the idle slots are exploited to duplicate the task. The proposed TDSA algorithm is compared with

Table 1: Analysis of existing works done by previous researchers.

Authors	Methodology used	Parameters addressed
Mustafa et al. [7]	ECTC	Energy consumption, migrations, SLA violations
Chen et al. [8]	OWC-A2C	Resource utilization, execution costs
Wang et al. [9]	IMPSO	Makespan, cost
Yao et al. [10]	TDSA	Makespan, resource utilization
Zhang et al. [11]	RMREC	Energy consumption
Malik et al. [12]	PSO based scheduler	Makespan, energy consumption, load balancing
Anitha and Vidyaraj [13]	Clustering approach	Prediction of SLA violation, workload
Faragardi et al. [14]	GRP-HEFT	Makespan
Aktan and Bulut [15]	DESA	Task completion time, load balancing
Hu et al. [16]	MOACO, CCA	CPU utilization, response time
Yao et al. [10]	TDSA	Makespan
Cui and Xiaoqing [17]	Two-dimensional coding	Execution costs
Sohani and Jain [25]	PMHEFT	Makespan, efficiency, power consumption
Mustafa et al. [26]	SLAAEERM	Energy efficiency, SLA compliance.
Talouki et al. [18]	TDA	Makespan, speedup
Talouki et al. [19]	Hybrid GA	Makespan
Shirvani and Talouki [20]	HH-LiSch	SLR, makespan, speedup
Shirvani [21]	HDPSO	SLR, speedup, efficiency
Tanha et al. [22]	TSAA	SLR, speedup, makespan
Shirvani and Talouki [23]	Bi-objective scheduling	Monetary cost, speedup, makespan
Alaie et al. [24]	HDCSA	SLR, speedup, efficiency

GRP-HEFT, greedy resource provisioning and modified heterogeneous earliest finish time; TDSA, task driven Scheduling algorithm; SLR, scheduling length ratio; IMPSO, immune mutation particle swarm optimization; CCA, cheetah chase algorithm; MOACO; modified ant colony optimization.

different baseline algorithms and decreases the makespan by up to 17.4%. The researchers have not focused on uncertainty in the workflow [10]. The researchers tried to minimize the makespan of workflow-based scheduling by reducing execution costs under deadline and budget constraints. The proposed algorithm is based on a GA. They used the top-down leveling method to assign priority to the task. They developed a two-dimensional coding method. The method is used to produce different offspring to increase the population's diversity in the cloud. They reduced the workflow scheduling costs [17]. Talouki et al. [18] proposed a task scheduling approach that addresses makespan, scheduling length ratio (SLR), and speedup. This approach uses heterogeneous resources and prioritized task mechanisms in upward and downward optimistic cost tables. Entire experimentation was carried out on workflows. It compared baseline algorithms, and the proposed approach dominated over existing algorithms. A hybridized approach, i.e., genetic and simulated annealing algorithms used to address makespan for effective workflow scheduling in the cloud environment [19]. All extensive simulations are conducted on workflowsim and proposed approach compared to existing approaches and improved parameters with this approach. A metaheuristic approach was developed to tackle heterogeneous resources in scheduling in the cloud paradigm aimed at minimization of makespan [20]. The entire scheduling pattern is based on task priorities and the insertion of tasks into appropriate slots based on task duplication. It evaluated over real-time workflows and identified that HH-LiSch outperforms existing algorithms for the specified parameter. The workflow scheduling approach is used to tackle scheduling in the cloud paradigm [21]. Discrete PSO is used as a methodology in this paradigm by generating an initial swarm smartly. It is evaluated over existing approaches and improvised parameters, i.e., SLR and efficiency. An effective task scheduling algorithm modeled using a hybridized approach, i.e., GA, simulated annealing algorithms [22]. This hybridized approach provides a balance between exploration and exploitation. It evaluated over state of the art algorithms and addressed parameters, i.e., SLR and speedup. A bi-objective task scheduling algorithm was developed using simulated annealing [23]. This approach improves scheduling patterns over HEFT and compared over realtime workflows and improvised parameters SLR, monetary cost, makespan, and efficiency. A bi-objective task scheduling approach was developed to address makespan and reliability [24]. This approach was developed and modeled using a discrete cuckoo search which balances between local and global search. It compared against existing approaches and addressed mentioned parameters.

In Table 1, the resource allocation security with efficient task scheduling in cloud computing-hybrid machine learning (RATS-HM) methodology is used,parameters addressed are resource utilization, energy consumption, and response time [27]. we analyzed various task and workflow scheduling algorithms that consider single and multiobjective approaches, but as scheduling in cloud computing is still a challenge as it is an NP-hard problem. Moreover, that prioritized workflow scheduling by minimizing energy consumption, SLA violation is not formulated by existing authors. Therefore, to bridge this gap and to tackle these parameters while scheduling workflows effectively onto virtual resources, we

formulated our proposed multiobjective workflow scheduling model by using cuckoo search optimization.

3. Mathematical Modelling

This section of our paper is based on the targeted outcomes of our research work. This section will elaborate on the major issues and challenges based on which we developed the proposed algorithm. To find out energy consumption and SLA violations, we consider the following parameters.

(a) Makespan

This section of our paper is based on the targeted outcomes of our research work. This section will elaborate on the major issues and challenges based on which we developed the proposed algorithm. We consider the following parameters to find out energy consumption and SLA violations.

$$\mathrm{Max}_i = \begin{cases} i = n \\ \mathrm{Fin}_i - \mathrm{St}_i, \\ i = 1 \end{cases} \quad (1)$$

where Fin_i = finish time of task in and St_i = start time of task i.

(b) Energy consumption

This parameter is used to calculate the total energy consumption of the servers available in a data center. It can be calculated using Equation (1).

Energy consumption: There are two types of energy consumption in cloud computing. Fixed energy consumption (E_f) and dynamic energy consumption (E_d). Fixed energy consumption due to the server being idle and dynamic energy consumption is related to energy consumed while any task is performed and other infrastructures are used in the cloud [26].

$$E_{\mathrm{total}} = E_{\mathrm{fixed}} + E_{\mathrm{dynamic}}. \quad (2)$$

Dynamic energy consumption is mainly based on cooling system (E_{col}), communication resource (E_{com}), storage resources (E_{st}), and computational energy (E_{cpu})

$$E_{\mathrm{dynamic}} = E_{\mathrm{col}} + E_{\mathrm{com}} + E_{\mathrm{st}} + E_{\mathrm{cpu}}, \quad (3)$$

$$E_{\mathrm{total}} = E_{\mathrm{fixed}} + E_{\mathrm{col}} + E_{\mathrm{com}} + E_{\mathrm{st}} + E_{\mathrm{cpu}}. \quad (4)$$

Energy consumption can be further divided based on idle time and active time. So if t is the idle time of the system, we may calculate CPU power consumption as follows:

$$E_{\mathrm{cpu}} = t \times P_{\mathrm{max}} + (1 - t) \times \mathrm{CPU}_{\mathrm{uti}}. \quad (5)$$

$$E_{\mathrm{comm}} = t \times \mathrm{Comm}_{\mathrm{max}} + (1 - t) \times \mathrm{Comm}_{\mathrm{uti}}. \quad (6)$$

$$E_{\mathrm{st}} = t \times \mathrm{St}_{\mathrm{max}} + (1 - t) \times \mathrm{St}_{\mathrm{uti}}. \quad (7)$$

Putting all in one we get total energy consumption in the data center

$$E_{\mathrm{total}} = \sum_{k=0}^{n} E_{\mathrm{fixed}} + t \times \{\mathrm{Mem}_{\mathrm{max}} + \mathrm{CPU}_{\mathrm{max}} + \mathrm{Comm}_{\mathrm{max}} + \mathrm{St}_{\mathrm{max}}\} + (1 - t)\{\mathrm{Mem}_{\mathrm{max}} + \mathrm{CPU}_{\mathrm{max}} + \mathrm{Comm}_{\mathrm{max}} + \mathrm{St}_{\mathrm{max}}\}. \quad (8)$$

Objective function is to minimize the power consumption so

$$\min \sum_{k=0}^{n} E_{\mathrm{fixed}} + t \times \{\mathrm{Mem}_{\mathrm{max}} + \mathrm{CPU}_{\mathrm{max}} + \mathrm{Comm}_{\mathrm{max}} + \mathrm{St}_{\mathrm{max}}\} + (1 - t)\{\mathrm{Mem}_{\mathrm{max}} + \mathrm{CPU}_{\mathrm{max}} + \mathrm{Comm}_{\mathrm{max}} + \mathrm{St}_{\mathrm{max}}\}. \quad (9)$$

Power consumption will be at a minimum when the idle time is 0. Idle energy consumption can be minimized by using dynamic voltage and frequency scaling of the particular core processor.

(c) SLA violation

The objective of this parameter is to find out the percentage of SLA violations by comparing the expected execution time with the actual execution time. An SLA violation is the ratio of the total demand to the allocated resources, main memory, storage, and bandwidth. SLA violation in all tasks taking into account the CPU we will receive.

$$\mathrm{SLA} = \sum_{k=0}^{n} \frac{A_{\mathrm{CPU}} - E_{\mathrm{CPU}}}{A_{\mathrm{mem}}}. \quad (10)$$

In this research, to expedite task scheduling process, initially, we calculate the priorities of tasks using the length of

Table 2: Example of task inputs.

Task	Priority	CPU time (ms)	RAM time (ms)	Storage time (ms)	Bandwidth (kbps)
T_1	6	23	2	5	12
T_2	1	10	6	6	3
T_3	1	9	7	10	5
T_4	5	5	8	22	9
T_5	3	3	12	12	2
T_6	2	23	19	23	6
T_7	4	21	12	45	4
T_8	4	11	4	5	8
T_9	1	11	55	12	3
T_{10}	2	12	12	12	10

task and processing capacity of VMs. It is calculated using Equation (11).

$$T^{\text{prio}} = \frac{T_k^{\text{size}}}{\text{VM}_n^{\text{pro}}}. \tag{11}$$

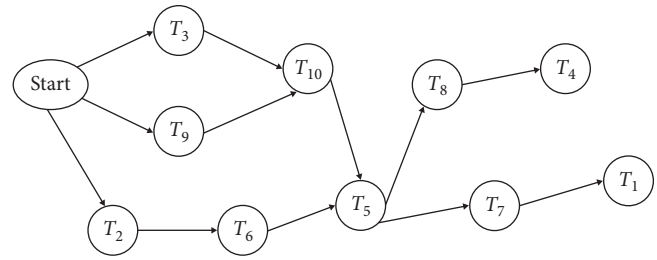

Figure 1: A sample DAG with 10 tasks and five depth levels.

4. Proposed Algorithm

Cloud providers have a major role in cloud computing as they need to manage maximum objectives with minimum resources. On the other hand, cloud users focus on the performance of the resources within a minimum time span [25]. A workflow application in cloud computing consists of a set of tasks with the dependency of some tasks on other tasks, like a parent–child task, where the child task cannot be executed until the parent task gets executed. These dependencies make the execution of tasks more difficult. Our proposed algorithm is based on task prioritization based on SLA violation and the expected time required to complete the work. This is possible only if we get the VMs that are more efficient at the time the tasks are entered into the process. Again, it is necessary to know the possible SLA violation in terms of percentage, so the algorithm will find out the exact VM where the task should be executed. This will reduce SLA violations as well as makespan and overall energy consumption.

In our proposed algorithm, we have assumed there are n number of VMs as $\{\text{VM}_1,\text{VM}_2,\ldots,\text{VM}_n\}$, k numbers of tasks as $\{T_1,T_2,\ldots,T_k\}$, with i number of PMs as $\{P_1,P_2,\ldots,P_i\}$ and number of j datacentres represented as $\{D_1,D_2,\ldots,D_j\}$. The tasks are dependent on other tasks and on the resources on which they will be used. The tasks are grouped into different levels depending on the dependency. Within the group, the tasks are prioritized in order to minimize SLA violations. The tasks with maximum length and processing capacity are assigned with higher priorities within the group/level. The independent task will be executed first based on the priorities. To handle the SLA violation and reduce energy consumption and the makespan, it is necessary to identify the VM where the execution will take less time and without any SLA violations. To identify the best VM, we used a cuckoo search algorithm. The cuckoo search algorithm will give the best VMs after the number of random walks like a cuckoo.

4.1. Problem Formulation. We precisely defined workflow scheduling problem by considering a workflow, i.e., directed acyclic graph (DAG) as $g = (v,e)$, i.e., v indicates a set of tasks, and they are represented as $\{T_1,T_2,\ldots,T_k\}$ and shown in Table 2. In the DAG, e indicates dependency, which is called interdependency in workflows. These workflows are running on v^n virtual resources, and they are indicated as $\{\text{VM}_1,\text{VM}_2,\ldots,\text{VM}_n\}$ and in turn these virtual resources reside in physical resources indicated as $\{P_1,P_2,\ldots,P_i\}$ and in turn physical resources are sitting in datacentres, and they are indicated as $\{D_1,D_2,\ldots,D_j\}$. From the above statement, problem can be defined in a way that T_k interdependent tasks to be scheduled on to VM_n virtual resources to address the percentage of SLA violation, makespan, and energy consumption.

When the tasks enter the system, the global system checks the priority and dependency of the task. According to the task priority and workflow dependency, the system will rearrange the tasks and generate the new DAG [18], and it is shown in Figure 1. When a new task is added to the system, the global broker regenerates the DAG based on the available tasks and task dependencies. This process will continue till there are no more tasks to perform.

We used the cuckoo search algorithm to find the most efficient VMs based on the available resources for the VMs. This will help to push the task into the VM, where it can perform its tasks without any SLA violation. This also saves energy and decreases the execution time. We used a nature-inspired

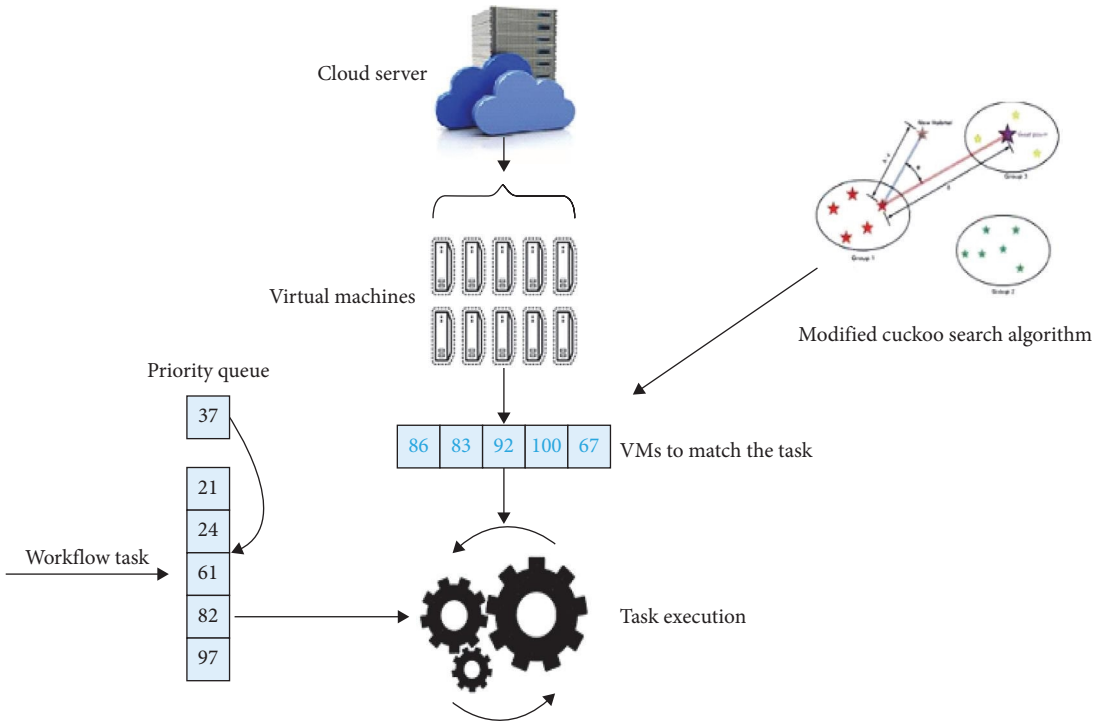

Figure 2: Proposed system architecture.

multiheuristic cuckoo search algorithm. Xin-She and Suash Deb developed the algorithm based on the brooding parasitism of some cuckoo species and the laying strategy. This algorithm will give us the most optimum VM, which is required in our workflow scheduling. The algorithm is based on cuckoo breeding behavior and levy flights. The proposed architecture is shown in Figure 2, and the flow of execution of the algorithm is shown in Figure 3.

A random walk called "levy flights" is used for the step lengths, and it is distributed according to a probability distribution.

4.2. Calculation of Time Complexity of Proposed Approach. Initially, in Algorithms 1–3 all the k tasks are received from various heterogeneous cloud users, and for all tasks priorities are calculated, and time complexity to calculate all tasks priorities is denoted as $O(k)$. After the evaluation of priorities, all these tasks are to be fed to scheduler. Therefore, time complexity to fed tasks to scheduler is $O(m)$, and after the collection of priorities, it is the responsibility of scheduler to generate schedules onto n VMs by considering those priorities, and its time complexity is $O(n)$. Therefore, overall time complexity is to be represented as $O(k + m + n)$.

5. Experimental Setup

The performance of our algorithm is evaluated using a workflow framework. It supports simulations and modeling of large-scale cloud environments with cloudlets as jobs. The jobs are predefined in the montage.xml file with their

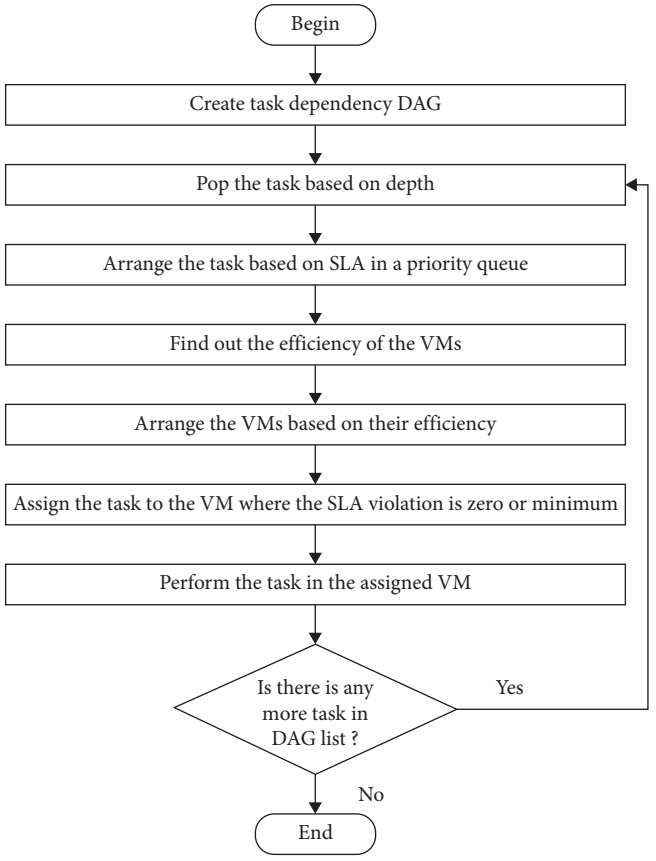

Figure 3: Flow of execution of workflow tasks.

Input: All task at given time
Output: Prioritized DAG
Step-1: Initialize Task $T = \{T_1, T_2, T_3, \ldots, T_i\}$
Step-2: For each task T_i in T at a given time
Step-3: Find the Task with highest priorities using Equation (11) and require minimum Resource Time
Step-4: Push the Task in the Queue
Step-5: Identify the task with Priorities and minimum dependency
Step-6: Push the Task in the Queue
Step-7: Is there is any more task continue to step 2
Step-8: Return the final DAG

Algorithm 1: (Prioritization of Task).

Input: Ms with their speed, bandwidth, RAM capacity
Output: Arrange the VMs based on the efficiency
Step 1: Input the VMs
Step 2: Generate the initial population
Step 3: continue till $t <$ max generation or satisfy the stop criteria
Step 4: Get the Cuckoo randomly by using the levy fights
Step 5: Evaluate the Cuckoo fitness
Step 6: Choose a nest randomly
Step 7: if $F_i > F_j$ then
Step 8: Replace F_j by the new solution
Step 9: end if
Step 10: Remove the worst nests and build the new nests
Step 11: go to step 2
Step 12: use genetic algorithm based cross over and mutation technique to find out the solution
Step 13: Return the order of the VMs

Algorithm 2: (Modified cuckoo search algorithm).

Input: All task in the DAG
Output: Makespan, SLA Violation and Total Energy Consumption
Step-1: Initialize the task
Step-2: Wait for any task
Step-3: If any new tasks fix it at the appropriate position in the DAG and generate the new DAG based on Algorithm 1
Step-4: Find out the VM with minimum or zero SLA violation
Step-5: Calculate the VM efficiency using modified cuckoo search algorithm
Step-6: Execute the task
Step-7: If any new task goes to step 3
Step-8: Find out makespan, SLA violation and Energy Consumption

Algorithm 3: (Task Scheduling and Execution).

dependencies. There are two categories of files, with 25 workloads and 100 workloads. Based on the study of the existing algorithms, we defined a set of values as the target SLA violation. This set is fixed for all the algorithms we used to get the violations in our testing. The result of makespan, SLA violation, and the energy consumption is effective because the values are fixed to all algorithms and compared for all algorithms. We extend the existing work by using prioritized task scheduling. The tasks are entered into the system based on their dependencies, and they are placed in the DAG at different levels. The tasks are entered into the process only based on their level, i.e., the most independent jobs are entered into the system first. Then the systems will be arranged based on the order of minimum time of completion

Table 3: Configuration settings for simulation.

Name of entity	Quantity
Tasks	25–100
Length of dependent tasks	600,000
VMs	5–15
Memory of physical host	16 GB
Storage capacity of physical host	8 TB
Bandwidth of physical host	1,200 Mbps
Memory of VMs	2 GB
Hypervisor	Xen
Operating system	MAC
No. of datacentres	15

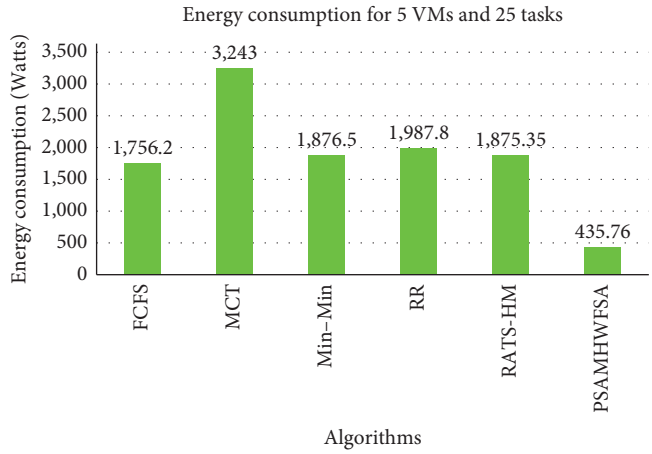

Figure 4: Calculation of energy consumption for 5 VMs and 25 tasks.

to minimize the SLA violation. Then, in the second phase, the tasks will be assigned to the VM, which will complete the job quickly. As a result, the best-fit VM will be assigned to the cloud task. As a result, there will be minimal wastage of resources, and the resources will be used for other processes.

We have used a different number of VMs and different numbers of tasks to test the validity of our proposed algorithm. We performed six levels of testing with 5, 10, and 15 VMs, as well as 25 and 100 cloudlets with work dependency. The result of our algorithm (PSAMHWFSA) was tested against the most popular and common algorithms like Max–Min, first-come, first-served (FCFS), minimum completion time (MCT), Min–Min, RATS-HM, and Round Robin. We compare the results for power consumption, makespan, and SLA violations. Our observations clearly show that it decreases SLA violations and makespan. At the same time, it reduces the power consumption of the system.

Each time a job is entered into the system, it will be placed in the DAG. When the job comes to the system from different clients, first, they will be sorted based on the task dependency. Then the system pulls the tasks and checks the SLA of each task. Once the process is over, it will find the most efficient VMs to complete the job. This process will continue for all the tasks.

Entire simulation was conducted on workflowsim, and we generated DAG workflow from the simulator randomly by considering all the abovementioned in this section and detailed configuration settings for our simulation represented in Table 3.

6. Simulations and Results

We have used workflow simulator software to test our algorithm (PSAMHWFSA) against the existing algorithms. We develop our algorithm based on a scheduling algorithm based on a GA like a cuckoo search algorithm and an execution plan. We used a data center with a different set of hosts for our experiments, with setups like 10 hosts and 20 VMs in each. The jobs are entered into the system randomly. Our proposed algorithm will check the job dependency and find the jobs based on the priority. The top-most priority job will be executed first. Similarly, our algorithm will find the best VMs based on the available resources to minimize power consumption. This paper compares our algorithm based on makespan, SLA violation, and energy consumption. The mutually exclusive tasks are placed on one level so that they can be executed in parallel. Our algorithm will accept all the tasks one level at a time. As our prime objective is to minimize SLA violations and energy consumption, our proposed algorithm checks each level's SLA, makespan, and energy consumption and executes the tasks based on it. Our proposed algorithm checked the VM with an efficiency suite to the task where the violation is zero or close to zero. This will minimize the SLA violation and keep the most efficient resources free for other tasks.

6.1. Calculation of Energy Consumption Using Different Scenarios. The system calculates the power consumption of the system for processing the task based on CPU time, RAM used, and bandwidth used in the system. Once the process is over, the system will display the overall power consumption of the system. We calculate the energy consumption of the VMs based on static and dynamic power consumption. We test it with a different set of VMs and tasks. The experimental results show that our algorithm outperforms well-known and common algorithms.

Initially, for the calculation of energy consumption, we considered 5 VMs and 25 tasks and workload generated randomly from workflowsim. Generated energy consumptions for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, RATS-HM, and PSAMHWFSA are 1,756.2, 3,243, 1,876.5, 1,987.8, 1,875.35, and 435.76, respectively. From Figure 4, it is evident that the proposed approach outperforms existing algorithms in view of energy consumption for the above scenario.

For the calculation of energy consumption, we considered 10 VMs and 25 tasks and workload generated randomly from workflowsim. Generated energy consumptions for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, RATS-HM, and PSAMHWFSA are 2,976.3, 1,123.5, 1,089.7, 5,323, 2,133.4, and 335.87, respectively. From Figure 5, it is evident that the proposed approach outperforms existing algorithms in view of energy consumption for the above scenario.

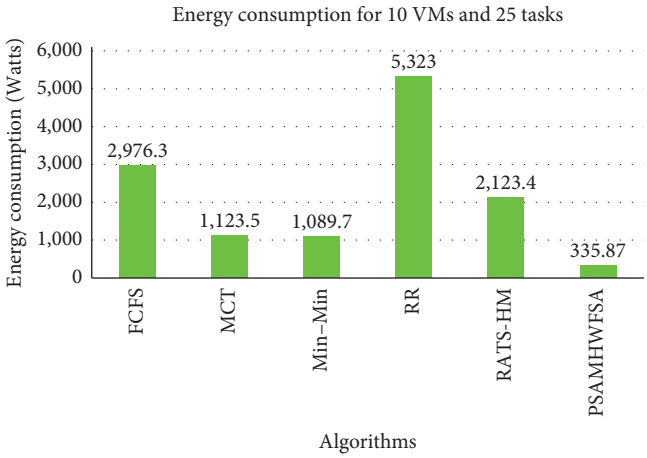

Figure 5: Calculation of energy consumption for 10 VMs and 25 tasks.

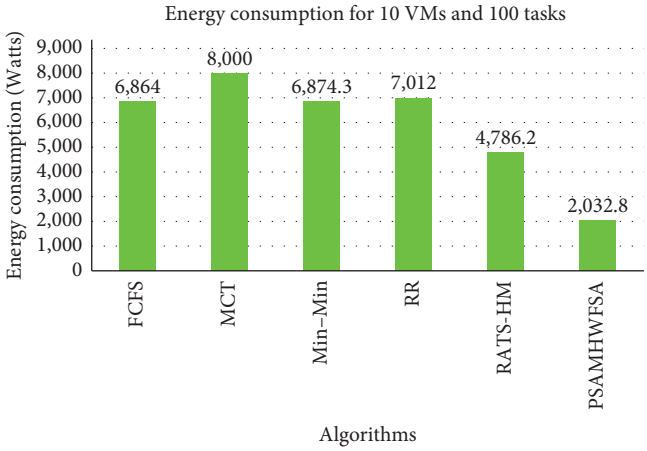

Figure 6: Calculation of energy consumption for 10 VMs and 100 tasks.

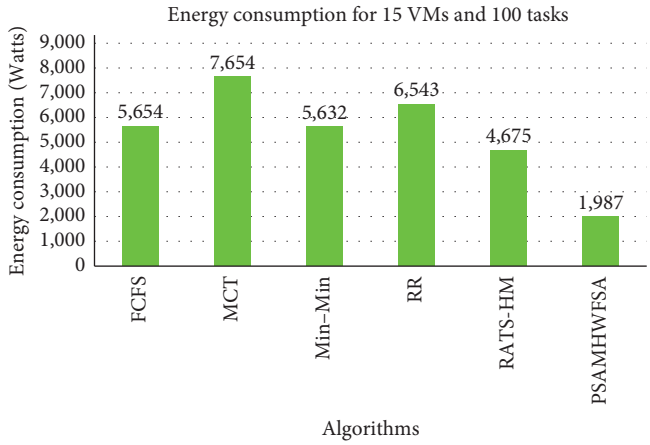

Figure 7: Calculation of energy consumption for 15 VMs and 100 tasks.

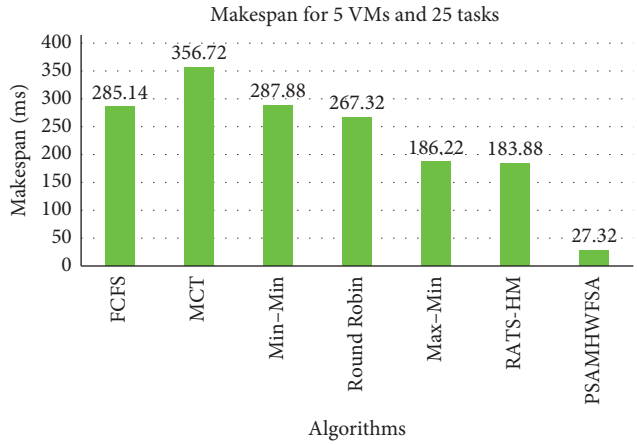

Figure 8: Calculation of makespan for 5 VMs and 25 tasks.

For the calculation of energy consumption, we considered 10 VMs and 100 tasks and workload generated randomly from workflowsim. Generated energy consumptions for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, RATS-HM, and PSAMHWFSA are 6,864, 8,000, 6,874.3, 7,012, 4,786.2, and 2,032.8, respectively. From Figure 6, it is evident that the proposed approach outperforms existing algorithms in view of energy consumption for the above scenario.

For the calculation of energy consumption, we considered 15 VMs and 100 tasks and workload generated randomly from workflowsim. Generated energy consumptions for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, RATS-HM, and PSAMHWFSA are 5,654, 7,654, 5,632, 6,543, 4,675, and 1,987, respectively. From Figure 7, it is evident that the proposed approach outperforms existing algorithms in view of energy consumption for the above scenario.

6.2. Calculation of Makespan Using Different Scenarios. It is used to calculate the maximum time taken to complete the entire task entered into the system. It determines the maximum time of all the VMs used in the process. We calculate the tasks' makespan based on each task's entry time and exit time against each VM. Then find out the time which is the maximum among the VMs. This will give the maximum time taken to complete the tasks. We test it with a different set of VMs and tasks. Our experimental results and the outcomes show that our proposed algorithm outperforms the well-known and common algorithms.

Initially, for the calculation of makespan, we considered 5 VMs and 25 tasks and workload generated randomly from workflowsim. Generated makespan for abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 285.14, 356.72, 287.88, 267.32, 186.22, 183.88, and 27.32, respectively. From Figure 8, it is evident that the proposed approach outperforms existing algorithms in view of makespan for the above scenario.

For the calculation of makespan, we considered 10 VMs and 25 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 276.87, 387.23, 298.12, 245.36,

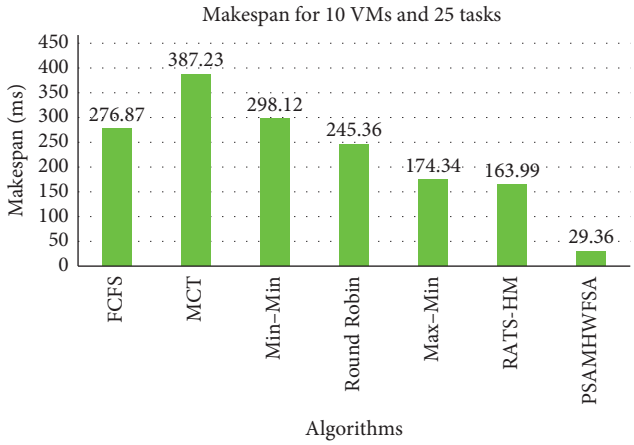

Figure 9: Calculation of makespan for 10 VMs and 25 tasks.

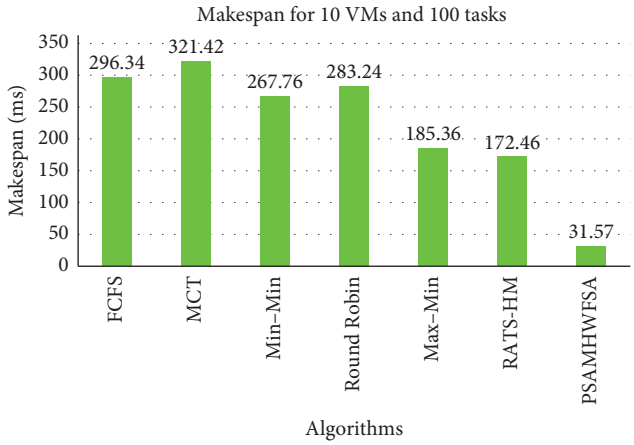

Figure 10: Calculation of makespan for 10 VMs and 100 tasks.

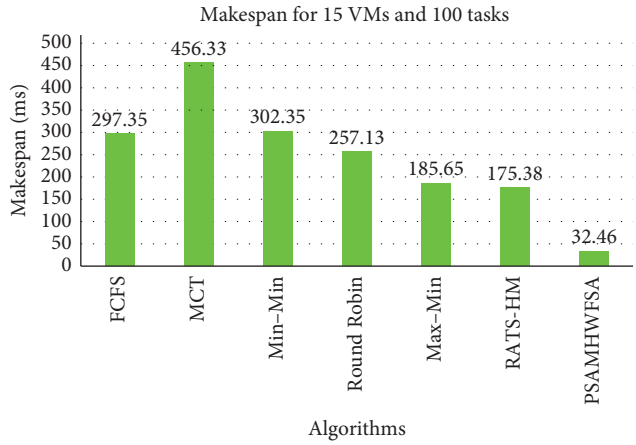

Figure 11: Calculation of makespan for 15 VMs and 100 tasks.

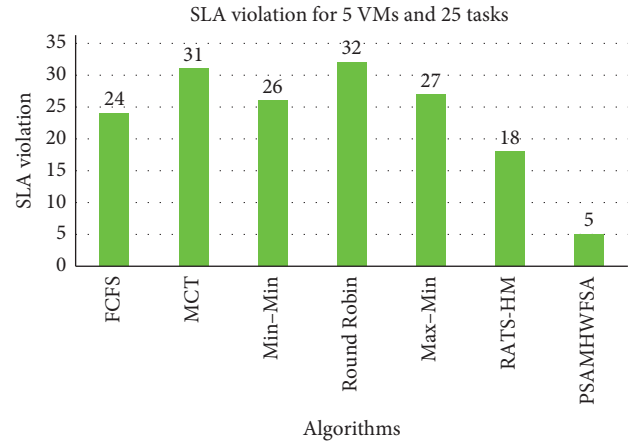

Figure 12: Calculation of SLA violation for 5 VMs and 25 tasks.

174.34, 163.99, and 29.36, respectively. From Figure 9, it is evident that the proposed approach outperforms existing algorithms in view of makespan for the above scenario.

For the calculation of makespan, we considered 10 VMs and 100 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 296.34, 321.42, 267.76, 283.24, 185.36, 172.46, and 31.57, respectively. From Figure 10, it is evident that the proposed approach outperforms existing algorithms in view of makespan for the above scenario.

For the calculation of makespan, we considered 15 VMs and 100 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 297.35, 456.33, 302.35, 257.13, 185.65, 175.38, and 32.46, respectively. From Figure 11, it is evident that the proposed approach outperforms existing algorithms in view of makespan for the above scenario.

6.3. Calculation of SLA Violation Using Different Scenarios. This policy verifies the SLA violation of each task based on a predefined SLA. Before any task is performed, the SLA violation will be checked, and the VM will be used to perform the task where the violation is zero or minimum. Each time a task is entered into the process, it will find its corresponding SLA as per the agreement. Then it checks the idle VMs where the task can be performed without violations or minimum violations. We used the cuckoo search algorithm to find the most efficient VMs.

Then the system finds the one where there is a minimum or zero violation. As per the output, it is clear that our proposed algorithm outperforms the well-known and common algorithms.

Initially, for the calculation of SLA violation, we considered 5 VMs and 25 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 24, 31, 26, 32, 27, 18, and 5, respectively. From Figure 12, it is evident that the proposed approach outperforms existing algorithms in view of SLA violations for the above scenario.

For the calculation of SLA violation, we considered 10 VMs and 25 tasks and workload generated randomly from workflowsim. Generated makespans for the

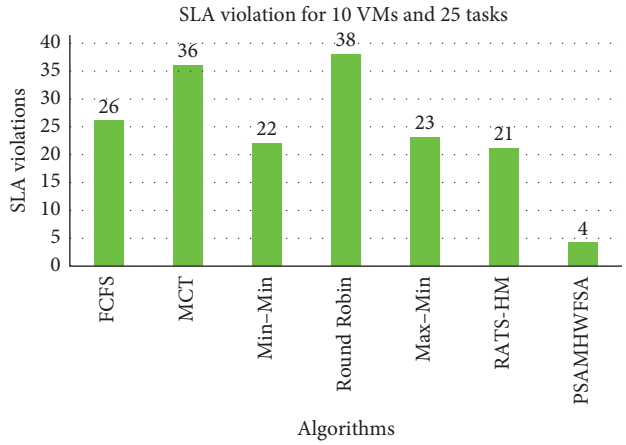

Figure 13: Calculation of SLA violation for 10 VMs and 25 tasks.

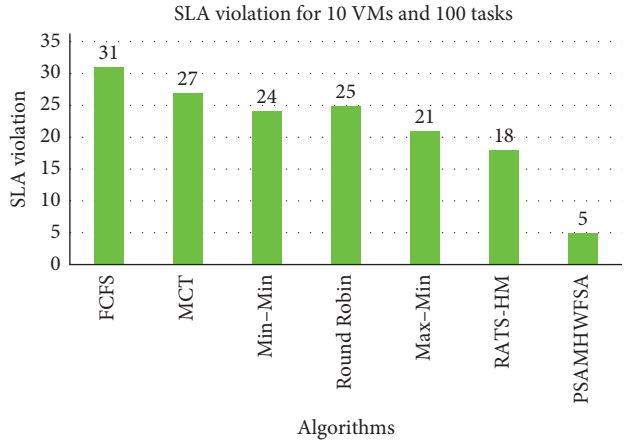

Figure 14: Calculation of SLA violation for 10 VMs and 100 tasks.

abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 26, 36, 22, 38, 23, 21, and 4, respectively. From Figure 13, it is evident that the proposed approach outperforms existing algorithms in view of SLA violations for the above scenario.

For the calculation of SLA violation, we considered 10 VMs and 100 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 31, 27, 24, 25, 21, 18, and 5, respectively. From Figure 14, it is evident that the proposed approach outperforms existing algorithms in view of SLA violations for the above scenario.

For the calculation of SLA violation, we considered 10 VMs and 100 tasks and workload generated randomly from workflowsim. Generated makespans for the abovementioned VMs and tasks for FCFS, MCT, Min–Min, RR, Max–Min, RATS-HM, and PSAMHWFSA are 26, 29, 25, 31, 26, 19, and 3, respectively. From Figure 15, it is evident that the proposed approach outperforms existing algorithms in view of SLA violations for the above scenario.

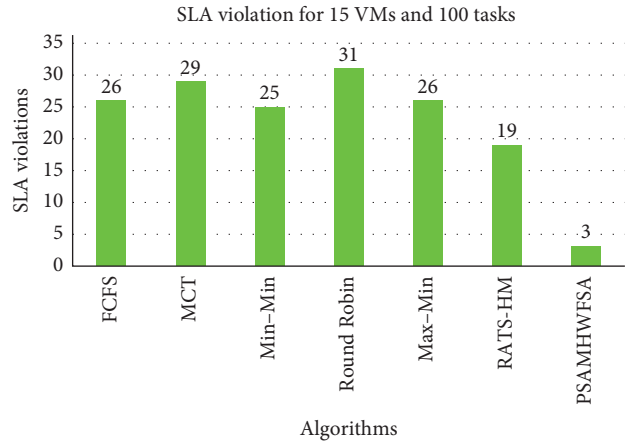

Figure 15: Calculation of SLA violation for 15 VMs and 100 tasks.

7. Conclusion and Future Works

The algorithm we proposed is based on workflow issues in cloud computing. We used three different algorithms to determine the task priorities, VM performance mapping, and finally, calculate the makespan, energy consumption, and SLA violation. We used DAG representation to represent the different tasks based on the time required to complete them. The time required is based on CPU, memory usage, storage, and bandwidth required time. We simulate the algorithms using the workflow simulator. We compare our proposed algorithm with the popular workflow-based algorithms. Our proposed approach is modeled by calculating the priorities of tasks, and then it fed to scheduler to generate schedules on to precised VMs while minimizing SLA violations and energy consumption. It was compared over various scheduling algorithms, i.e., Max–Min, FCFS, MCT, Min–Min, RATS-HM, and Round Robin, and the proposed approach outperforms for the mentioned parameters, i.e., SLA violation by 22% and energy consumption by 15%. Our proposed algorithm can be enhanced further to minimize the migration time. Our proposed algorithm can be enhanced further to minimize the migration time. This research used a metaheuristic approach, but still, the scheduling problem in the cloud paradigm is a highly dynamic situation, and with this approach, it is not able to predict the type of upcoming workload onto the cloud console. Therefore, In future work, we want to employ a machine learning technique to schedule tasks effectively based on workload arriving onto cloud console by predicting the type of worklogs that comes onto the cloud paradigm.

Data Availability

The data can be provided by the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] B. Gul, I. A. Khan, S. Mustafa, O. Khalid, S. S. Hussain, and D. Dancey, "CPU and RAM energy-based SLA-aware workload consolidation techniques for clouds," *IEEE Access*, vol. 8, pp. 62990–63003, 2020.

[2] E. I. Elsedimy and F. Algarni, "Toward enhancing the energy efficiency and minimizing the SLA violations in cloud data centers," *Applied Computational Intelligence and Soft Computing*, vol. 2021, Article ID 8892734, 14 pages, 2021.

[3] N. D. Vahed, M. Ghobaei-Arani, and A. Souri, "Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: a comprehensive review," *International Journal of Communication Systems*, vol. 32, no. 14, Article ID e4068, 2019.

[4] F. Fakhfakh, H. H. Kacem, and A. H. Kacem, "Workflow scheduling in cloud computing: a survey," in *2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, pp. 372–378, IEEE, Ulm, Germany, 2014.

[5] Y. Gao, S. Zhang, and J. Zhou, "A hybrid algorithm for multi-objective scientific workflow scheduling in IaaS cloud," *IEEE Access*, vol. 7, pp. 125783–125795, 2019.

[6] Y. Wang, H. Liu, W. Zheng et al., "Multi-objective workflow scheduling with Deep-Q-Network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

[7] S. Mustafa, K. Sattar, J. Shuja et al., "SLA-aware best fit decreasing techniques for workload consolidation in clouds," *IEEE Access*, vol. 7, pp. 135256–135267, 2019.

[8] Z. Chen, K. Lin, B. Lin, X. Chen, X. Zheng, and C. Rong, "Adaptive resource allocation and consolidation for scientific workflow scheduling in multi-cloud environments," *IEEE Access*, vol. 8, pp. 190173–190183, 2020.

[9] P. Wang, Y. Lei, P. R. Agbedanu, and Z. Zhang, "Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm," *IEEE Access*, vol. 8, pp. 29281–29290, 2020.

[10] F. Yao, C. Pu, and Z. Zhang, "Task duplication-based scheduling algorithm for budget-constrained workflows in cloud computing," *IEEE Access*, vol. 9, pp. 37262–37272, 2021.

[11] L. Zhang, L. Wang, Z. Wen, M. Xiao, and J. Man, "Minimizing energy consumption scheduling algorithm of workflows with cost budget constraint on heterogeneous cloud computing systems," *IEEE Access*, vol. 8, pp. 205099–205110, 2020.

[12] N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, "Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds," *Applied Sciences*, vol. 11, no. 13, Article ID 5849, 2021.

[13] R. Anitha and C. Vidyaraj, "Workload and SLA violation prediction in cloud computing," in *2019 Third International Conference on Inventive Systems and Control (ICISC)*, pp. 582–587, IEEE, Coimbatore, India, 2019.

[14] H. R. Faragardi, M. R. S. Sedghpour, S. Fazliahmadi, T. Fahringer, and N. Rasouli, "GRP-HEFT: a budget-constrained resource provisioning scheme for workflow scheduling in IaaS clouds," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 6, pp. 1239–1254, 2020.

[15] M. N. Aktan and H. Bulut, "Metaheuristic task scheduling algorithms for cloud computing environments," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 9, Article ID e6513, 2022.

[16] Y. Hu, H. Wang, and W. Ma, "Intelligent cloud workflow management and scheduling method for big data applications," *Journal of Cloud Computing*, vol. 9, Article ID 39, 2020.

[17] Y. Cui and Z. Xiaoqing, "Workflow tasks scheduling optimization based on genetic algorithm in clouds," in *IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 6–10, IEEE, Chengdu, China, 2018.

[18] R. N. Talouki, M. H. Shirvani, and H. Motameni, "A heuristic-based task scheduling algorithm for scientific workflows in heterogeneous cloud computing platforms," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, Part A, pp. 4902–4913, 2022.

[19] R. N. Talouki, M. H. Shirvani, and H. Motameni, "A hybrid meta-heuristic scheduler algorithm for optimization of workflow scheduling in cloud heterogeneous computing environment," *Journal of Engineering, Design and Technology*, vol. 20, no. 6, pp. 1581–1605, 2022.

[20] M. H. Shirvani and R. N. Talouki, "A novel hybrid heuristic-based list scheduling algorithm in heterogeneous cloud computing environment for makespan optimization," *Parallel Computing*, vol. 108, Article ID 102828, 2021.

[21] M. H. Shirvani, "A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems," *Engineering Applications of Artificial Intelligence*, vol. 90, Article ID 103501, 2020.

[22] M. Tanha, M. H. Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments," *Neural Computing and Applications*, vol. 33, pp. 16951–16984, 2021.

[23] M. H. Shirvani and R. N. Talouki, "Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with *makespan* and monetary cost minimization approach," *Complex & Intelligent Systems*, vol. 8, pp. 1085–1114, 2022.

[24] Y. A. Alaie, M. H. Shirvani, and A. M. Rahmani, "A hybrid bi-objective scheduling algorithm for execution of scientific workflows on cloud platforms with execution time and reliability approach," *The Journal of Supercomputing*, vol. 79, pp. 1451–1503, 2023.

[25] M. Sohani and S. C. Jain, "A predictive priority-based dynamic resource provisioning scheme with load balancing in heterogeneous cloud computing," *IEEE Access*, vol. 9, pp. 62653–62664, 2021.

[26] S. Mustafa, K. Bilal, S. U. R. Malik, and S. A. Madani, "SLA-aware energy efficient resource management for cloud environments," *IEEE Access*, vol. 6, pp. 15004–15020, 2018.

[27] P. K. Bal, S. K. Mohapatra, T. K. Das, K. Srinivasan, and Y.-C. Hu, "A joint resource allocation, security with efficient task scheduling in cloud computing using hybrid machine learning techniques," *Sensors*, vol. 22, no. 3, Article ID 1242, 2022.