Energy-Aware Scheduling of Tasks in Cloud Computing

Yamina Mehor*, Mohammed Rebbah and Omar Smail
Computer Science Department, University of Mustapha Stambouli Mascara, Mascara, Algeria
E-mail: yamina.mehor@univ-mascara.dz, rebbahmed@univ-mascara.dz, o.smail@univ-mascara.dz
*Corresponding author

Keywords: cloud computing, task scheduling, energy consumption, adaptive genetic algorithm, SLA violation, execution time

Received: February 20, 2024

Cloud computing Infrastructures have been created to facilitate consumer's access to various services through the Internet. Massive energy consumption by data centers that hosts Cloud applications result in high carbon footprints to the environment. Therefore, it is required to develop ways that reduce the energy consumption. These aspects are reduced by efficiently task scheduling within the deadline respect and providing the resources according to the user's request. Energy usage, execution time, and SLA violations in virtualized cloud data centers are discussed in this study. For effective scheduling, the suggested approach is predicated on job categorization and thresholds. Tasks having lengthy execution duration are preprocessed in the first stage by being placed in different lists. The following stage involves classifying tasks according to the resources required. Finally, Genetic Algorithm is used to select the best schedules. To represent the dynamic nature of the cloud environment and to offer a scheduling solution that is nearly optimum and decrease energy consumption, execution time, and SLA violation, an adaptive Genetic Algorithm is developed. By the use of cloud infrastructure simulation and a series of performance and quality assessment experiments, the suggested model is tested in this setting. Results show that the suggested method improves performance by reducing execution time, energy usage, and SLA violations.

Povzetek: Analizirano je energetsko učinkovito načrtovanje opravil v oblaku z uporabo prilagodljivega genetskega algoritma za zmanjšanje porabe energije, časa izvedbe in kršitev SLA. Model izboljša učinkovitost z dvostopenjskim pristopom k obdelavi in optimizaciji, kar so potrdili tudi rezultati simulacij.

1 Introduction

Cloud computing has emerged as a key paradigm in the world of computing. It contributes to the increasing expectations for availability and flexibility. Users of the Internet and computers are becoming more interested in the services proposed by the cloud computing providers due to its impressive growth in recent years[1]. Energy consumption is a crucial topic in cloud computing that has become a significant issue. It requires appropriate solutions and several data centers contain servers, cooling systems, switching and network components that make up the cloud computing infrastructure.[2] The energy consumed by data centers has increased due to the rising demand for infrastructure that has become a serious problem. Higher expenses of profit and CO2 emissions result from the excessive energy used. Therefore, efficient solutions are required to reduce the negative effects on the environment and cloud provider profit. Every year, energy cost rises and several studies have examined how much energy is needed by data centers and individual servers [3]. Numerous studies have been launched on the subject of energy and power in computing systems. The creation of virtual machine (VMs) within a physical server is made possible by virtualization technology that also enables to utilize resources more efficiently while using less hardware [4]. Task scheduling and energy efficiency are two key obstacles in resource allocation[5]. This paper presents an Energy-Aware Scheduling Model (EASM) for task scheduling in cloud computing. The objective of the proposed model is to reduce the energy consumption, execution time, and SLA violation. EASM works in two phases, i.e., pre-processing and optimization with Adaptive Genetic Algorithm. In the first phase, tasks with longer execution times are allocated in VMs with high processing capabilities[6]. In the next phase, GA is used to optimize scheduling and find better solutions. In the popular meta-heuristic method known as the genetic algorithm, populations of potential candidate solutions, known as individuals, are developed over many generations to find the best solution for a specific problem. With the contribution of various genetic operations, the optimization begins with random individuals and eventually reaches the global optimum [7]. The simulations' results confirm that the suggested approach is more robust and efficient in terms of energy usage, execution time, and SLA violations. The structure of this paper is as follows: We evaluate related work in section 2. We present the proposed model that aims to reduce energy in cloud computing in section 3. Then, we discuss in Section 4 the performance evaluation and experimentation. Finally, Section 5 concludes the paper.

2 Related Work

For load balancing, energy efficiency, and better resource scheduling, an effective cloud environment, (Feng, H. et al.) [8] using a variety of optimization algorithms, including the whale optimization algorithm (WOA), cat swarm optimization (CSO), cuckoo search algorithm (CSA), BAT, and particle swarm optimization (PSO) is created. The suggested work employs a cost-effective solution to the load balancing and resource scheduling issues.

(Ibrahim, H. et al.) [2] have selected two advanced scheduling algorithms to examine the outcomes in a same cloud computing environment and examine the approaches that maximize energy and cost in a cloud computing environment. The main objective of the energy-efficient strategy (EES) is to spread out the maximum load over the fewest possible virtual machines. By assigning appropriate resources to the required tasks, Cost-based Scheduling using Genetic Algorithm minimizes execution time that decreases user costs. The results are then studied and compared to other scheduling algorithms, such as Round-Robin (RR) and First-come- First-served (FCFS).

(Thekkepuryil, J.K.V. et al.) [9] present an Integer Linear Programming (ILP) model for cloud computing energy optimization and an Adaptive Genetic Algorithm (GA) for dynamic work scheduling in the cloud data center. In order to account for the dynamic nature of the cloud environment and to offer a near-optimal scheduling solution that reduces energy usage, an Adaptive Genetic Algorithm (GA) is developed. By allocating incoming tasks to resources in a way that both user needs and the energy consumption of cloud data centers are fulfilled. This study attempts to establish a model and an algorithm for reducing the energy consumption in a cloud computing infrastructure. It concentrates on a single Cloud data center as its environment settings.

(Medara, R. et al.)[10], authors suggest to use an energy-aware workflow scheduling technique for cloud computing with VM consolidation. The suggested EASVMC technique is designed to achieve many objectives including resource usage, VM migrations, and energy consumption. Task scheduling and VM consolidation are the two stages of the EASVMC algorithm's operation (VMC). The virtual machine that will consume the least amount of energy during the first phase is assigned to the task with the longest possible execution time. The second phase includes a well-known NP-hard issue, namely VM consolidation. Based on CPU utilization, the VMC phase divides the physical hosts into hosts with a regular load, under-loaded hosts, and overloaded hosts. Therefore, double threshold values are employed. Migration of virtual machines from overloaded and underloaded hosts to normally loaded hosts. Authors used the Water Wave Optimization (WWO) algorithm, a nature-inspired meta-heuristic approach, for the VMC phase. This algorithm finds an appropriate migration plan to reduce energy consumption by increasing overall resource utilization and switching off idle hosts after migrating their VMs to an appropriate target host. They evaluated the effectiveness of this algorithm in comparison to three well-known methods: HEFT, EES, and PESVMC. The simulation results demonstrated that the EASVMC algorithms surpass the other three techniques in terms of overall performance.

To overcome the drawbacks of task consolidation and scheduling, (Panda ,S.K. et al.) [11] proposed an energy-efficient task scheduling algorithm (ETSA). The proposed algorithm uses a normalization process to determine when to schedule tasks while taking into consideration their completion times and resource use overall. The energy efficient task-scheduling algorithm that is presented for reducing energy consumption and execution time is the foundation of this study. For heterogeneous cloud computing systems, the authors created an online energy-efficient work scheduling system. The proposed system can be used for cloud, application, energy, and scheduling models. In order to decide on scheduling, the method computes the completion time and overall resource usage of a job on the resources.

(Rajkumar Choudhary et al.) [12] suggested a new approach based on multi-objective optimization. For complicated VM scheduling solutions, they calculate the amount of energy used, the CPU usage, and the number of instructions performed in each scheduling period. Multi-objective PSO (particle swarm optimization) optimization can lead to better and more efficient results for various parameters than multi-objective GA (genetic algorithm) optimization in terms of energy efficiency and execution time reduction.

(Shaimaa Badr et al.) [13] focused on the issue of power consumption and proposes a powerful method called Task Consolidation based Power Minimization (TCPM). It effectively allocates jobs to the cloud environment's available resources in order to reduce power consumption. The best-fit approach is employed to achieve the optimum resource usage and prevent energy waste in the proposed TCPM algorithm that improves and incorporates various advantages of the current algorithms. The results of the proposed TCPM algorithm are compared with FCFS, WWO, and MCT algorithms using the CloudSim toolkit.

(Nimra Malik et al.) [6] suggested a method for effective scheduling and improved resource usage based on task categorization and thresholds. Workflow tasks are preprocessed in the first stage to prevent bottlenecks by separating tasks with high dependencies and lengthy execution durations. The following phase is classifying tasks according to the intensity of the resources needed. To choose the optimum schedules, Particle Swarm Optimization (PSO) is employed. To verify the suggested approach, experiments were done. Comparative results from benchmark datasets are given. The findings demonstrate how the suggested algorithm performs better than the other algorithms in terms of energy usage, execution time, and load balancing.

(Sasan Gharehpasha et al.) [14] developed a novel method for optimum placement of virtual machines utilizing a combination of the Sine-Cosine and Salp Swarm algorithms as discrete multi-objective and chaotic functions. The initial objective of the suggested method was to decrease the amount of electricity used in cloud data centers

by reducing the quantity of physically active devices. The second objective was to decrease resource waste and control it by strategically placing virtual machines on actual equipment in cloud data centers. The third goal was to keep Service Level Agreement amongst the active physical computers in cloud data centers to a minimum. By using the suggested approach, the migration of virtual machines onto real equipment is prevented from growing. In the end, the suggested algorithm's results were compared with the results of First Fit, Modified Best Fit Decreasing, and Virtual Machine Placement Ant Colony System.

In [15], in order to schedule the workflow tasks to the VMs and dynamically deploy/undeploy the VMs in accordance with the workflow task's needs, an energy and resource efficient workflow scheduling algorithm (ERES) is presented. To determine the EC of the servers, an energy model is offered. It uses a double threshold strategy to determine if the server is overloaded, underloaded, or operating normally. Live VM migration is used to balance the load on the overloaded/underloaded servers. Live VM migration strategy is used. Extensive simulation tests are run to evaluate the efficacy of the suggested approach. On the basis of RU, energy efficiency, and task execution time, the suggested approach is compared to the PESVMC (power efficient scheduling and VM consolidation) algorithm. Additionally, the results are validated in a genuine cloud environment. The outcomes show how successful the suggested ERES algorithm is.

(Shishidoa,H.,Y. et al.) [3] investigated the effectiveness of using meta-heuristic techniques for scheduling cloud processes. The purpose of this study was to evaluate the effects of GA and PSO augmentation on workflow scheduling optimization. To assess the competency of the meta-heuristic technique, a cost-aware workflow scheduling issue was used. PSO, GA, and Multi Population GA meta-heuristics were also used in the experiments. The evaluation of meta-heuristic algorithms was based on the objectives of cost minimization and time for response. These algorithms produced more effective schedules that reduce costs in a reasonable amount of time.

The proposal of [16] presents a multi-objective optimization method for cloudlet computing that makes use of the non-dominated sorting idea. The objectives taken into consideration include delay, user energy consumption, cloudlet energy consumption, and cost, which are determined by the number of cloudlets. Non-dominated sorting genetic algorithms (NSGA-III and NSGA-II) are employed to be compared to this proposed work.

In [17], authors offer a task scheduling heuristic for heterogeneous cloud systems that saves energy. It performs by choosing the best physical host with virtual machines while taking into account the utilization of any incoming tasks on that specific virtual machine. They demonstrate the superiority of the proposed heuristic in energy-efficient task scheduling in heterogeneous cloud settings by comparing its energy efficiency with other previous methods, including ECTC, MaxUtil, Random, and FCFS, on both synthetic and benchmark datasets.

The authors in [18] provide a new hybrid method for effective virtual machine placement that combines the Sine Cosine Algorithm (SCA) with the Ant Colony Optimization (ACO) algorithm. The results obtained by the ACO algorithm have been examined using SCA, an advancing search method that makes use of the Sine and Cosine functions in the engineering domain.The ACO method has been utilized to exploit the search space's solutions for effective virtual machine placement, hence facilitating power management and reducing resource wastage.

The table 1 illustrates a summary that compares the reviewed approaches in terms of key performance metrics, such as energy efficiency, execution time, and SLA violation rates.

3 The proposed model

In this section, discussion of the system model and energy model is followed by the details of each phase.

3.1 System Model

Scheduling is the process of allocating a number of tasks to a number of resources (virtual machines). In the cloud data centers, there are two levels of scheduling: (i) series of rules for deploying VMs at the server level and (ii) rules for assigning tasks to VMs. The main focus of this paper is VM-level task scheduling techniques. The scheduling approach is a strategy for selecting which resources to use to execute tasks in order to shorten execution times and conserve energy.

Consider the Cloud Data Center (CDC) consists of N physical machines (PM). It can be represented in Eq. (1):

$$CDC = \{PM_1, PM_2, ..., PM_N\} \tag{1}$$

where PM_i (i=1,…,N) denotes the PMs presented in the CDC.The features of PM_i are defined in Eq. (2):

$$PM_i = \{C_i, Size\_PM_i, RAM\_PM_i, Bandwidth\_PM_i, \#Core\_PM_i\} \tag{2}$$

Consider the physical machine consists of M virtual machines (VMs) It can be represented as in Eq. (3):

$$PM_{ij} = \{VM_{i1}, VM_{i2}, ..., VM_{iM}\} \tag{3}$$

where (j=1,….,M) M is the number of virtual machines obtained from PM_i. The features of VM are defined in Eq. (4):

$$VM_{ij} = \{c_{ij}, Size\_VM_{ij}, RAM\_VM_{ij}, Bandwidth\_vm_{ij}, \#Core_{ij}\} \tag{4}$$

Year	Approche	Energy consumption	Execution time	SLA violation
2021	Feng, H. et al.[8]	Yes	Yes	No
2018	Ibrahim, H. et al.[2]	Yes	Yes	No
2021	Thekkepuryil, J.K.V. et al.[9]	Yes	Yes	Yes
2021	Medara, R. et al.[10]	Yes	Yes	No
2019	Panda, S.K.[11]	Yes	Yes	No
2022	Rajkumar Choudhary et al. [12]	Yes	Yes	No
2022	Shaimaa Badr et al.[13]	Yes	Yes	Yes
2021	Nimra Malik et al.[6]	Yes	Yes	No
2021	Sasan Gharehpasha et al. [14]	Yes	No	Yes
2021	Neha Garg et al.[15]	Yes	Yes	Yes
2018	Shishido, H., Y. et al.[3]	No	Yes	Yes
2023	Ali Salah Alasady et al.[16]	Yes	No	Yes
2021	Mahendra Kumar Gourisaria et al.[17]	Yes	Yes	No
2024	C.Vijaya et al. [18]	Yes	Yes	Yes

Table 1: summary table

The tasks submitted by the users can be represented as in Eq. (5):

$$Tasks = Tasks\_D \cup Tasks\_O \tag{5}$$

Where Tasks_D is set of tasks submitted by users with the consideration of deadline constraints. Tasks_O is set of tasks submitted by users without the consideration of deadline constraints.

$$Tasks\_D_{kd} = \{TD_1, TD_2, ..., TD_P\} \tag{6}$$

where P is the number of tasks submitted with the consideration of deadline constraints.

$$Tasks\_O_{ko} = \{TO_1, TO_2, ..., TO_L\} \tag{7}$$

where L is the number of tasks submitted without consideration deadline constraints. The features of Tasks_D and Tasks_O are defined in Eq. (8) and (9):

$$TD_{kd} = \{length_{kd}, FileSize_{kd}, Deadline_{kd}\} \tag{8}$$

$$TO_{ko} = \{length_{ko}, FileSize_{ko}\} \tag{9}$$

3.2 Energy Model

The processing capacity c_{ij} of a resource VM_{ij} is computed with the MIPS of each VM. The capacity of M VMs is calculated with Eq. (10)[6].

$$C_i = \sum_{j=1}^{M} c_{ij} \tag{10}$$

In cloud computing, resource use has a major effect on how much energy is used. The utilization can be calculated with Eq. (11)[6].

$$u_i = \frac{\sum_{j=1}^{M} c_{ij}}{C_i} \tag{11}$$

Where M is the number of VMs running on PM_i, and c_{ij} refers to the computing allocated to VM_{ij}. In Eq. (11), C_i is the total processing capacity of the PM_{ij}. This research examines CPU use that determines how much electricity physical devices consume. About 70% of the power of a physically active machine is used when it is inactive. So, using Eq. (12), the power consumption (u) as CPU utilization is defined as: [19]

$$P(u)_i = P_{max}(0,7 + 0,3{\times}u_i) \tag{12}$$

where u_i is the current CPU usage and P_{max} is the maximum power of a physical system operating at 100% CPU utilization. CPU usage is defined as a function u(t) of time since it varies over time. As a result, Eq. (13) establishes a physical machine's (PM_i) total energy consumption: [19]

$$E_i = \int f(u(t))\,\mathrm{d}t \tag{13}$$

3.3 Scheduling model

The main objective of the suggested approach is to decrease the amount of energy used, the execution time, and SLA violations of the cloud resources while taking diverse users priorities into account and optimizing the energy and execution time under the deadlines constraints. This paper proposes a tasks scheduling model in cloud computing that treats two sets of tasks. The first set of tasks takes priority since the users require the deadline unlike the second set of tasks. The respect of deadline of first set will involve more energy consumption compared to the energy consumption of the second set. Two possible scenarios are distinguished,

Symbol	**Description**
CDC	Cloud Data Center.
PM_i	Physical machine. i = (1,…,N)
N	Number of physical machines in the cloud environment
$Size\_PM_i$	The size of PM_i.
RAM_PM_i	The RAM of PM_i.
$Bandwidth\_PM_i$	The Bandwidth of PM_i.
$\#Core\_PM_i$	Number of cores in PM_i.
VM_{ij}	Virtual machine. j=(1,…,M)
M	The number of virtual machines in the cloud environment.
$Size\_VM_{ij}$	The size of VM_{ij}.
RAM_VM_{ij}	The RAM of VM_{ij}.
$Bandwidth\_VM_{ij}$	The Bandwidth of VM_{ij}.
$\#Core\_VM_{ij}$	Number of cores in VM_{ij}.
$Tasks_k$	The tasks submitted by the users in DCD. k=(1,…,Ntsk)
Ntsk	The number of tasks submitted in the cloud environment, where Ntsk = L+P.
$Tasks\_D_{kd}$	Set of tasks with deadline constraints. Kd=(1,…,P)
P	The number of deadlined tasks submitted in the cloud environment.
$Tasks\_O_{ko}$	Set of tasks without deadline constraints. Ko=(1,…,L)
L	The number of no_deadlined tasks submitted in the cloud environment.
Td_{kd}	Deadlined Task.
TO_{ko}	No-deadlined task.
$Length_{kd}$	The length of deadlined task.
$FileSize_{kd}$	The size of deadlined task.
$Deadline_{kd}$	Time till which the tasks should be finished.
$Length_{ko}$	The length of no-deadlined task.
$FileSize_{ko}$	The size of no-deadlined task.
C_i	The total processing capacity of PM_i.
c_j	The processing capacity of VM_{ij}.
u_i	The current CPU utilization of PM_i.
P_i	The power of PM_i.
P_{max}	The maximum power of a physical machine.
E_i	The total energy consumption PM_i.
ECT_k	The required execution time of task on VM_{ij}.
$\%SLAviolation_i$	The percentage of tasks that have exceeded their deadlines in PM_i.

Table 2: Symbols used in the proposed method.

one for deadlined tasks and the other for no-deadlined ones. In these two cases, two phases are applied. In the first phase, thresholds are used for the tasks length. Tasks with longer execution times are allocated in VMs with high processing capabilities. Once the energy consumption reaches a threshold, the second phase will be launch. The genetic algorithm is a global scheduler that allocates incoming cloud tasks to suitable VMs. These two phases are used to reduce the execution time applying to decrease energy consumption as resources are utilized efficiently.

3.3.1 Task allocation phase

In the first phase [6], a new method that is proposed and intended to dynamically prioritize the tasks and schedule them to the best suitable selected resource. The tasks in Cloud Computing require to be executed by the available resources to achieve minimal total time for completion. The expected completion time for the task is defined in Eq. (14):[6]

$$ECT_k = \frac{Length_k}{c_{ij}}; k = 1, 2, ..., Ntsk; \\ i = 1, 2, 3, ..., N; j = 1, 2, 3, ..., M \quad (14)$$

Where ECT_k is the time needed for the kth task to execute on the Mth virtual machine, where M is the number of VMs and $Ntsk$ is the number of tasks.

$Length_k$ is the length of a task in Million Instruction (MI) and c_{ij} is the VM_{ij} speed Million Instructions Per Second (MIPS).

Allocation of tasks with deadline constraint

$$Tasks\_D_{kd} = \{TD_1, TD_2, ..., TD_P\} \quad (15)$$

Where P is the number of tasks submitted with the consideration of deadline constraints. Tasks_D is the first set of tasks requiring top priority processing compared to the second set of tasks to avoid SLA violation. The execution time ECT of each task should be less or equal then the deadline.

$$(ECT_{kd} \leq Deadline_{kd}) AND (Min(ECT_{kd})) \quad (16)$$

Task allocation depends on the length of each task and respecting the deadline constraint where the proposed algorithm sets a threshold for the length of the tasks and the threshold values are applied. They are intended to prioritize tasks during execution. To decrease the total execution time, the lengthier tasks need to be processed first. High processing capacity virtual machines are assigned to these tasks.

Allocation of tasks without deadline constraint

$$Tasks\_O_{ko} = \{TO_1, TO_2, ..., TO_L\} \quad (17)$$

where L is the number of tasks submitted without consideration of deadline constraints. Tasks_O is the second set of tasks requiring second priority processing.

$$Min(ECT_{ko}) \quad (18)$$

Task allocation depends on the length of each task where the proposed algorithm sets a threshold for the length of the tasks and the threshold values are used to prioritize tasks during execution. As a result, each task's priority is established according to its duration. Tasks with longer length need to be processed with priority and VMs with high processing capabilities are allocated to these tasks.

Algorithm 1 Task Allocation

Input: $TD1\_list, TO2\_list, VM\_list, c_{ij}, task\_length$

Output: Allocation(TK,VMij)

while $TD1 <> Null$ **do** Search_execMin_deadl (TD1.length, VM) // search high performance (VM) to the longest tasks and respect deadline constraint

while $TO2 <> Null$ **do** search_execMin (TO2.length, VM) //search high performance (VM) to the longest tasks.

Return Allocation (TK,VMij).

Algorithm 1 shows the steps involved in the procedure where this one is composed of two functions:
Search_execMin_deadl: the objective of this function is to search the available resources (VMs) for the deadline tasks in the data center to achieve Minimal total time for completion while respecting the deadline constraint of each task.
Search_execMin: seeks to search for the available resources (VMs) for the no-deadline tasks in the data Center to achieve Minimal total time for completion.

3.3.2 Task scheduling phase

In the second phase, the proposed algorithm uses two modified Genetic Algorithm (GA) to optimize scheduling and find better solutions for the two sets of tasks (with deadline and without deadline).

Encoding

In this paper, the choice is to use a direct representation; Table 3 illustrates the encoding representation. In the suggested example, there are two major information. The tasks that are scheduled and the number of the VM instance to which it is assigned are shown in table 3.[20]

Task ID	T1	T2	...	Tn-1	Tn
VM ID	VM1	VM3	...	VM2	VM4

Table 3: Encoding

Initial Population

The creation of an initial population of T-size solution candidates for evolution is the first stage in the optimization utilizing genetic algorithms process. T-size refers to the population size. Every population set has numerous chromosomes containing genes corresponding to various tasks

planned on distinct virtual machines.[7]. The one set of chromosomes is presented in table 4.

Task ID	T1	T2	T3	T4	T5
VM ID	VM1	VM3	VM2	VM2	VM3

Table 4: Initial Population

Fitness function:
Which chromosomes to pass on to the following generation depends critically on their level of fitness.
Fitness Function for set of tasks with deadline constraint:
The fitness value of deadlined tasks is defined in Eq. (19):

$$fitnessfunction01 = \alpha \sum_{i=1}^{N} E_i + \beta \sum_{i=1}^{N} \%SLAviolation_i, \quad \alpha + \beta = 1 \quad (19)$$

Where E_i refers to the energy consummated by the PM_i. $\%SLA\_violation$ refers to the percentage of tasks that have exceeded their deadlines in PM_i.
Fitness Function for set of tasks without deadline constraint is defined in Eq. (20):

$$fitnessfunction02 = \sum_{i=1}^{N} E_i \quad (20)$$

where E_i refers to the energy consummated by the PM_i. Therefore, the population's best and worst chromosomes have the lowest and highest fitness rates, respectively.
Selection operation
Populations are ranked according to their fitness levels, choosing the best elite chromosomes of a predefined size, and passing them on to the following generation.
Crossover
In GAs, the crossover operator is crucial for changing the population chromosomes. The crossover operator improve population evolution in GAs. The operator joins several chromosomes to form a new generation of chromosomes. While certain characteristics are inherited from both parents, others are inherited from one parent only[8]. The individuals from the previous stage are used in this study. They go through a process called crossover where genes are exchanged at random crossing points. As seen in tables 5,6,7 and 8, Chosen individuals will produce two offspring following a crossover.

Task ID	**T1**	T2	**T3**	T4	**T5**
VM ID	**VM1**	VM3	**VM2**	VM2	**VM3**

Table 5: Parent 1

Task ID	**T1**	T2	**T3**	T4	**T5**
VM ID	**VM3**	VM2	**VM1**	VM3	**VM2**

Table 6: Parent 2

Task ID	**T1**	T2	**T3**	T4	**T5**
VM ID	**VM3**	VM3	**VM1**	VM2	**VM2**

Table 7: Offspring 1

Task ID	**T1**	T2	**T3**	T4	**T5**
VM ID	**VM1**	VM2	**VM2**	VM3	**VM3**

Table 8: Offspring 2

Mutation
By changing chromosomes, mutations are used to maintain population variety. To create variation in the population, many chromosomes are mutated by the mutation operator after being combined using the combination operator. As indicated in tables 9 and 10, a random VM has been provided to a task from the list of tasks at random.[21]

Task ID	T1	T2	**T3**	T4	T5
VM ID	VM1	VM2	**VM2**	VM3	VM3

Table 9: Before Mutation

Task ID	T1	T2	**T3**	T4	T5
VM ID	VM1	VM2	**VM3**	VM3	VM3

Table 10: After Mutation

Termination conditions
For each objective function, the individual of each generation is compared to the previous best fitness value. If the new individual outperforms the old one, the best value is updated[7]. The suggested method ends when all of the chromosomes, or solutions, converge to the same degree of fit. However,there are no further improvements to the fitness value.[21]

4 Experimental evaluation

The experiments conducted to evaluate the suggested energy-aware scheduler are presented in this section. The evaluation configuration, comprising the cloud infrastructure, the scheduler algorithm, and the machine utilized to perform the scheduling, is discussed in the initial part of this section. The results of the scheduling for the various situations are shown in the second section. A series of experiments are completed to assess the effectiveness of the scheduling algorithm after analyzing the impact of different factors and algorithms on the execution time and energy consumed.

The research presents a unique method of work scheduling in cloud settings that is solely evaluated with the CloudSim simulator and makes use of modified genetic algorithms. This approach outperforms conventional heuristic techniques in terms of minimizing SLA violations, reducing execution time, and optimizing consumption of

energy. The efficacy of sophisticated algorithmic techniques for cloud infrastructure management is validated, that makes a substantial contribution to the theory of distributed systems. In practical terms, the approach can save energy costs and increase operational efficiency for cloud data centers, however it still has to be verified in real-world scenarios. Before a large-scale implantation, a phased approach that begins with controlled test settings is advisable. Cloud service providers may manage varying workloads with more efficiency and dependability by using this strategy.

4.1 Simulation experiments

In order to evaluate the proposed algorithms, the proposed solutions has been implemented using the CloudSim simulator. To carry out the experiments, an Intel i3-4005U CPU 1.70 GHz and 4 GB of memory have been utilized.

4.1.1 Cloud infrastructure

In this simulation experiments, one data center was created and contained a number of PMs. A variety of VMs types are created in this simulation environment. The specific parameters are listed in Table 11.

In this research, several important factors inform the choice of simulation settings. Firstly, representativeness is essential; the simulations are pertinent since the characteristics selected match common data center schemes based on previous studies. Second, to represent a wide range of system behaviors and situations, different Physical Machines (PMs), Virtual Machines (VMs), and tasks are chosen. Thirdly, the use of standard setups ensures repeatability, making it easier to compare the results with previous studies and increasing their validity.

This paper treats two sets of tasks. The first one takes priority since the users require the deadline unlike the second one. The respect of the deadline will involve more energy consumption compared to the energy consumption by the second set.

4.1.2 Scheduler configuration

After the submission of the tasks by the users, this study seeks to allocate the tasks with deadlines to the first. Then, we allocate the tasks without deadline. In this phase, we allocate each task to the fastest VM. Finally, we control the energy levels based on an energy threshold.

Threshold for energy consumption: Once the energy consumption reaches this threshold, we will launch the scheduling phase based on Genetic Algorithm.

Genetic Algorithm: Initialization and looping algorithms were divided into two categories. The optimal solution was identified by evaluating the fitness values after a random viable solution had been created during the initialization procedure. Subsequently, the looping segments confirmed if a certain terminal condition was satisfied. The mutation, crossover, and selection processes were used in order throughout the continuous loop. In the end, the process of iteration produced the optimum solution. [7][20]

A sensitivity study is conducted to evaluate the model's resilience under various conditions. This included analyzing workload intensity variations to comprehend how varying demand levels affected system performance. Also, modifications were examined to the VM and PM setups to determine how resource allocation influenced results. Additionally, a variety of resource management strategies were evaluated, focusing on how different work allocation techniques affected system performance. This thorough examination strengthened the model's validity and resilience, guaranteeing its dependability in a variety of situations.

4.1.3 Experimental results

The algorithms were evaluated in terms of execution time, energy consumption, and SLA violation. To confirm the effectiveness of the approaches over the ones already in use, an extensive statistical studies were conducted, including comparison tests. A statistically meaningful improvement is shown from the results.

In the simulation experiments, we compare the proposal with:

Naïve Genetic Algorithm (NGA): in this experiment, we allocate user tasks by FCFS technique and we use GA after reaching the energy threshold without difference between deadline and no-deadline tasks.

Round-Robin: we allocate deadline tasks by Round-Robin technique and the no-deadlined tasks with the FCFS technique. The proposal treats two priorities of two types of tasks that are deadlined and no-deadline tasks. The deadline tasks take the first priority to involve violations. In these two cases of tasks, two thresholds are proposed, one for tasks length and the other for energy consumed. Tasks with length longer than first threshold are allocated in VMs with high processing capabilities. Once the energy consumption reaches the second threshold, we will launch the genetic algorithm.

Execution time

First, we evaluated the performance of our algorithm by varying the number of tasks from 50 to 600.

Experiment 1: We changed the number of tasks as indicated in Fig. 1 and measured the performance efficiency using a fixed number of virtual machines (VMs) of 30.

Experiment 2:

As seen in Fig. 2, we set a limit of 30 tasks and a range of 10 to 50 virtual machines. Compared to the suggested approach, NGA and RR take longer in both scenarios to accomplish a task.

Experiment 3: In the third scenario, the number of VMs and tasks are not fixed as shown in Fig. 3.

Fig. 1, 2, and 3 show the comparative analysis of the execution time of set of algorithms. The three figures present the evaluation results of EASM vs. NGA and

Entity Type	Parameters	Values
Data Center	Number of Data Center	1
PM	Number of PM	50
	C (MIPS)	4000-8000
VM	Number of VM	10-60
	C(mips)	1000-4000
Tasks	Number of Tasks	10-10000
	Length (MI)	10000-30000

Table 11: The Resources Parameters.

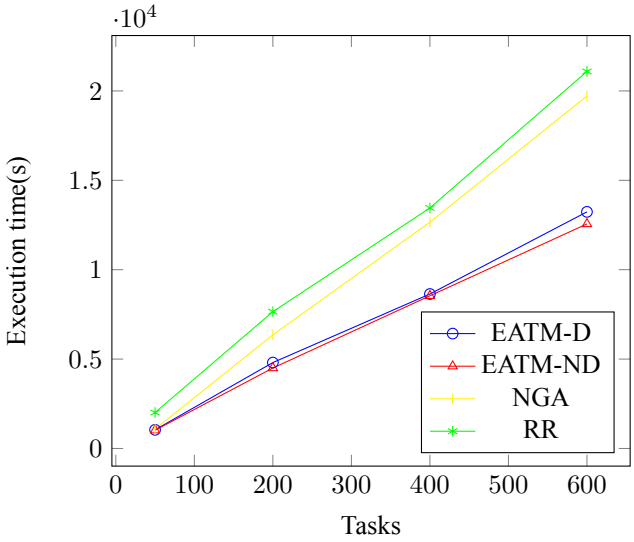

Figure 1: Execution time(s) of different numbers of deadlined and no-deadlined tasks.

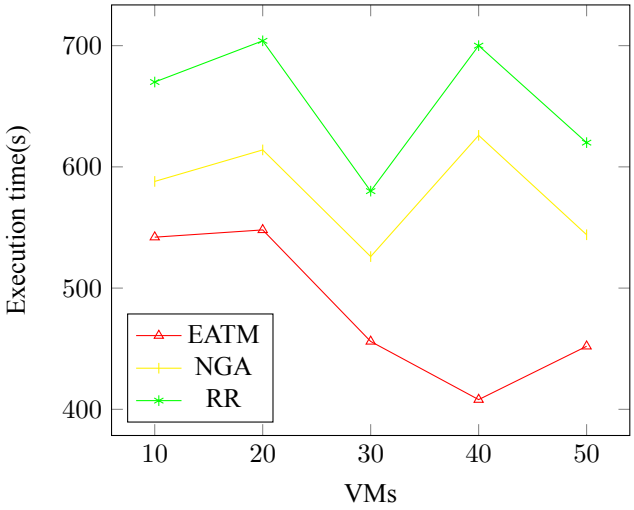

Figure 2: The Execution time(s) of different numbers of VMs.

RR. Fig. 1 presents the execution time regarding tasks number. Fig. 2 presents the execution time regarding VMs number and Fig. 3 presents the execution time in different experimentations. As shown, EASM outperforms NGA and RR. This is due to using the proposed algorithm that decreases the consumed time. The execution time of EASM is satisfying when compared with the NGA and RR since it is based on task classification and thresholds. The model has the potential to improve the execution time speed and optimization efficiency of the EASM. Even when the number of tasks rises, EASM has strong capacity to assess the outcomes attained, identify the greatest fitness value, and make the best decision. The respect of the deadline of the deadlined-tasks involves more execution time compared to the execution time of the no-deadlined ones.

Energy consumption

Now, we evaluated the performance EASM for energy consumption by varying the number of the tasks from 50 to 600.

Experiment 4: The fourth scenario of the experiments evaluated the energy consumption by the fixed number of VMs at 30 and a changed number of tasks as shown in Fig. 4.

Experiment 5:

As shown in Fig. 5, the tasks in this scenario are fixed at 30 and the range of virtual machines (VMs) is 10 to 50, increasing by 10.

Fig. 4 and 5 show the energy consumption comparison between deadlined and no-deadlined tasks of the proposed EASM, NGA and RR. The energy consumed by the EASM is considerably less than the NGA and RR. It is evident that there is a significant difference among the compared algorithms and EASM consumes less energy in different tasks. The respect of the deadline of the first set involves more energy consumption compared to the energy consumption of the second set of tasks.

Experiment 6: Energy consumption with the Changing of VMs and Tasks.

In this scenario, the number of VMs and tasks are not fixed as shown in Fig. 6.

Fig. 6 presents the energy consumption in different experimentation. Our EASM outperforms NGA and RR. This is due to using the proposed algorithm that decreases the consumed energy. The energy consumption of the proposed

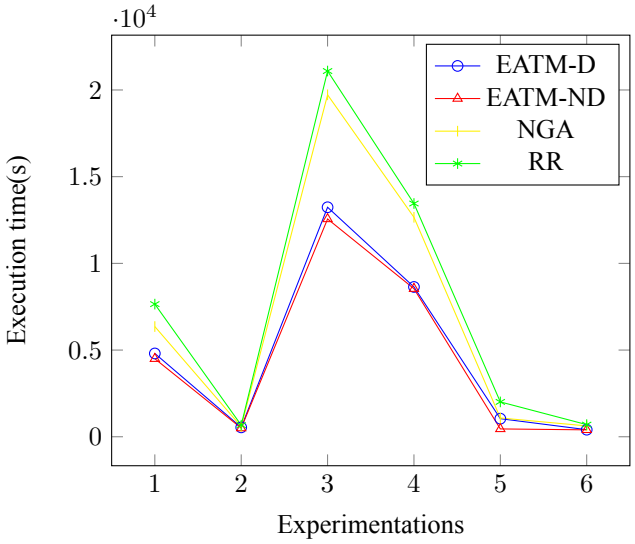

Figure 3: The Execution time(s) of different experimentations.

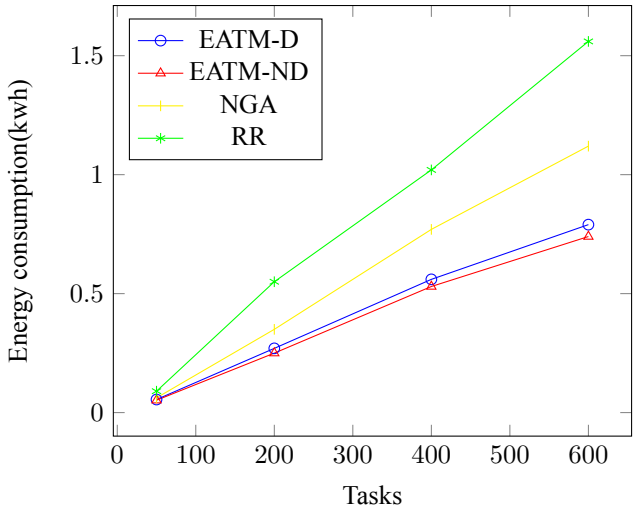

Figure 4: The energy consumption (Kwh) of different numbers of tasks.

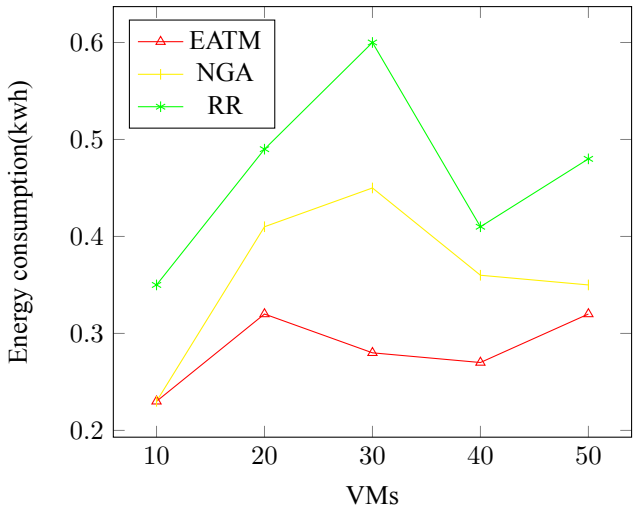

Figure 5: The energy consumption (Kwh) of different numbers of VMs.

algorithm is better when compared to the NGA and RR.

Experiment 7: The average SLA violation rate for all methods are shown in Fig. 7. It is evident that, in comparison to different approaches, EASM produced the lowest rate of SLA violations. The obtained results confirm the effectiveness of the model in minimizing SLA violations, due to using the task classification and thresholds.

The performance of EASM can be explained by the classification and priority mechanisms and algorithm searches for an optimal solution more quickly. This algorithm considers not only processing time and energy consumption, but also resource utilization and the number of resources that can effectively complete the user's task.

This study has resulted in several implications for cloud computing settings. Firstly, cloud service providers may be able to significantly minimize their operational costs as a result of the increased energy efficiency as well as shorter execution times. This would increase the financial viability of their offerings. Second, this method guarantees improved service quality by lowering Service Level Agreement (SLA) violation rates, that greatly raises customer satisfaction and trust in cloud services. Additionally, maintaining optimal performance in diverse settings and guaranteeing scalability and flexibility in the face of changing demands depend largely on the model's capacity to adjust to dynamic workload fluctuations in cloud data centers.

Although results are encouraging, this method has encountered several limitations. A significant constraint pertains to the results' generalization, as the experiments were carried out inside a simulated setting. In real-world, cloud settings must verify their validity so the findings are considered valuable. The scalability at large scale is another drawback. While being built for dynamic contexts, the model's effectiveness at extremely high scales still has to be carefully assessed to ensure it can manage complex and large-scale cloud infrastructures.

5 Conclusion and perspectives

Due to the size of cloud data centers, there is a significant energy consumption and longer task execution times. As a result, users must regularly transmit data and data centers use virtual machine scaling to improve the efficiency of system resources. The main purpose of this work is to schedule effectively work into the available cloud environment resources, minimal energy consumption, execution time, and SLA violation. Task categorization, thresholds, and queuing are the foundation of the proposed work. Tasks are gathered into queues in the first phase based on how long they will take to complete. Then, GA is applied to find better solutions to improve scheduling. The suggested model had been validated and the comparative experimental findings were presented in terms of execution time, energy efficiency, and SLA violation. The results demonstrated that

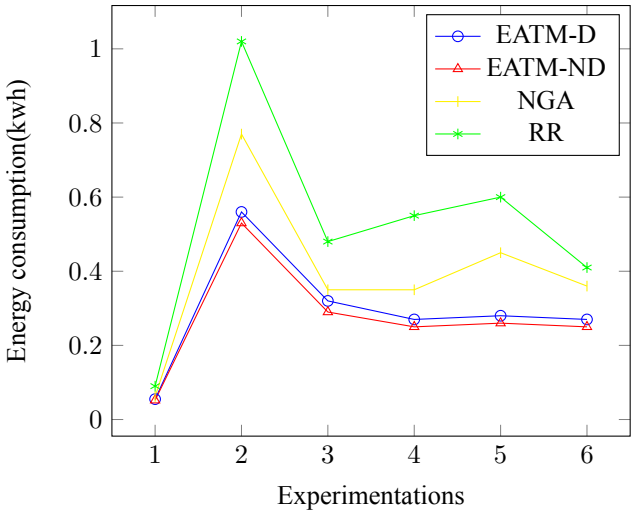

Figure 6: The energy consumption (Kwh) of different experimentation.

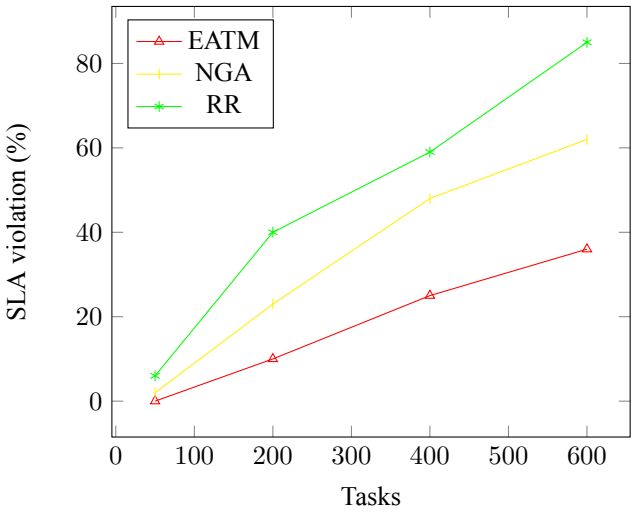

Figure 7: Average SLA violation of different tasks number.

for all parameters, the suggested algorithm surpassed the other approaches.

In future work, we concentrate on the stretch of the model by working on multiple data centers geographically distributed and we will improve the Multi-parameter energy functions making it possible to take into consideration all the factors that energy consume. There are other ways that might be investigated to advance this study. Extensive testing in real cloud systems is an essential step toward experimental validation, which will support and improve this model. Hybrid optimization is an extra approach that seems at different combinations of optimization methods in an effort to further enhance performance. Furthermore, for larger application, modifying the methodology is required for multi-cloud scenarios, in which resources are dispersed across several cloud service providers. Incorporating artificial intelligence and machine learning methods to predict workloads and dynamically modify resources might ultimately greatly improve the effectiveness and availability of cloud services.

References

[1] S. Goyal, S. Bhushan, Y. Kumar, A. U. H. S. Rana, M. R. Bhutta, M. F. Ijaz, and Y. Son, "An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm," *Sensors*, vol. 21, no. 5, pp. 1–24, 2021.

[2] H. Ibrahim, R. O. Aburukba, and K. El-Fakih, "An Integer Linear Programming model and Adaptive Genetic Algorithm approach to minimize energy consumption of Cloud computing data centers," *Computers and Electrical Engineering*, vol. 67, pp. 551–565, 2018. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2018.02.028

[3] H. Y. Shishido, J. C. Estrella, C. F. M. Toledo, and M. S. Arantes, "Genetic-based algorithms applied to a workflow scheduling algorithm with security and deadline constraints in clouds," *Computers and Electrical Engineering*, vol. 69, pp. 378–394, 2018. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2017.12.004

[4] R. Medara and R. S. Singh, "Energy Efficient and Reliability Aware Workflow Task Scheduling in Cloud Environment," *Wireless Personal Communications*, vol. 119, no. 2, pp. 1301–1320, 2021. [Online]. Available: https://doi.org/10.1007/s11277-021-08263-z

[5] H. A. Hassan, S. A. Salem, and E. M. Saad, "A smart energy and reliability aware scheduling algorithm for workflow execution in DVFS-enabled cloud environment," *Future Generation Computer Systems*, vol. 112, pp. 431–448, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2020.05.040

[6] 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 (Switzerland)*, vol. 11, no. 13, 2021.

[7] A. Nahhas, J. T. Cheyyanda, and K. Turowski, "An adaptive scheduling framework for the dynamic virtual machines placement to reduce energy consumption in cloud data centers," *Proceedings of the Annual Hawaii International Conference on System Sciences*, vol. 2020-January, no. January, pp. 878–887, 2021.

[8] H. Feng, Y. Deng, and J. Li, "A global-energy-aware virtual machine placement strategy for cloud data centers," *Journal of Systems Architecture*, vol. 116, no. July 2020, p. 102048, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2021.102048

[9] J. Kakkottakath Valappil Thekkepuryil, D. P. Suseelan, and P. M. Keerikkattil, "An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment," *Cluster Computing*, vol. 24, no. 3, pp. 2367–2384, 2021. [Online]. Available: https://doi.org/10.1007/s10586-021-03269-5

[10] R. Medara, R. S. Singh, and Amit, "Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization," *Simulation Modelling Practice and Theory*, vol. 110, no. December 2020, p. 102323, 2021. [Online]. Available: https://doi.org/10.1016/j.simpat.2021.102323

[11] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Cluster Computing*, vol. 22, no. 2, pp. 509–527, 2019. [Online]. Available: https://doi.org/10.1007/s10586-018-2858-8

[12] R. Choudhary and S. Perinpanayagam, "Applications of Virtual Machine Using Multi-Objective Optimization Scheduling Algorithm for Improving CPU Utilization and Energy Efficiency in Cloud Computing," 2022.

[13] S. Badr, A. El Mahalawy, G. Attiya, and A. A. Nasr, "Task consolidation based power consumption minimization in cloud computing environment," *Multimedia Tools and Applications*, 2022.

[14] S. Gharehpasha, M. Masdari, and A. Jafarian, "Power efficient virtual machine placement in cloud data centers with a discrete and chaotic hybrid optimization algorithm," *Cluster Computing*, vol. 24, no. 2, pp. 1293–1315, 2021. [Online]. Available: https://doi.org/10.1007/s10586-020-03187-y

[15] N. Garg, D. Singh, and M. S. Goraya, *Energy and resource efficient workflow scheduling in a virtualized cloud environment*. Springer US, 2021, vol. 24, no. 2. [Online]. Available: https://doi.org/10.1007/s10586-020-03149-4

[16] A. S. Alasady, W. A. Awadh, and M. S. Hashim, "Non-dominated sorting genetic optimization-based fog cloudlet computing for wireless metropolitan area networks," *Informatica (Slovenia)*, vol. 47, pp. 1–8, 2023.

[17] M. K. Gourisaria, P. M. Khilar, and S. S. Patra, "Esps: Energy saving power spectrum-aware scheduling to leverage differences in power ratings of physical hosts in datacenters," *Informatica (Slovenia)*, vol. 45, pp. 63–75, 2021.

[18] C. Vijaya and P. Srinivasan, "Multi-objective meta-heuristic technique for energy efficient virtual machine placement in cloud computing data centers," *Informatica (Slovenia)*, vol. 48, pp. 1–18, 3 2024.

[19] Y. Saadi and S. El, "Energy-efficient strategy for virtual machine consolidation in cloud environment," *Soft Computing*, vol. 2, 2020. [Online]. Available: https://doi.org/10.1007/s00500-020-04839-2

[20] H. Aziza and S. Krichen, "A hybrid genetic algorithm for scientific workflow scheduling in cloud environment," *Neural Computing and Applications*, vol. 32, no. 18, pp. 15 263–15 278, 2020. [Online]. Available: https://doi.org/10.1007/s00521-020-04878-8

[21] Z. Peng, P. Pirozmand, M. Motevalli, and A. Esmaeili, "Genetic Algorithm-Based Task Scheduling in Cloud Computing Using MapReduce Framework," *Mathematical Problems in Engineering*, vol. 2022, 2022.