6-23-2021

Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds

Nimra Malik
COMSATS University Islamabad

Muhammad Sardaraz
COMSATS University Islamabad

Muhammad Tahir
COMSATS University Islamabad

Babar Shah
Zayed University, babar.shah@zu.ac.ae

Gohar Ali
Sur University College

See next page for additional authors

Recommended Citation

Malik, Nimra; Sardaraz, Muhammad; Tahir, Muhammad; Shah, Babar; Ali, Gohar; and Moreira, Fernando, "Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds" (2021). *All Works*. 4347.
https://zuscholars.zu.ac.ae/works/4347

applied sciences

Article

Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds

Nimra Malik [1], Muhammad Sardaraz [1,*], Muhammad Tahir [1], Babar Shah [2], Gohar Ali [3] and Fernando Moreira [4]

[1] Department of Computer Science, Attock Campus, COMSATS University Islamabad, Attock 43600, Pakistan; sp19-rcs-012@cuiatk.edu.pk (N.M.); m_tahir@cuiatk.edu.pk (M.T.)
[2] College of Technological Innovation, Zayed University, Abu Dhabi 144534, United Arab Emirates; babar.shah@zu.ac.ae
[3] Department of Information Systems and Technology, Sur University College, Sur 411, Oman; goharali@suc.edu.om
[4] REMIT, Universidade Portucalense, 4200-072 Porto, Portugal; fmoreira@uportu.pt
* Correspondence: sardaraz@cuiatk.edu.pk

Abstract: Cloud computing is a rapidly growing technology that has been implemented in various fields in recent years, such as business, research, industry, and computing. Cloud computing provides different services over the internet, thus eliminating the need for personalized hardware and other resources. Cloud computing environments face some challenges in terms of resource utilization, energy efficiency, heterogeneous resources, etc. Tasks scheduling and virtual machines (VMs) are used as consolidation techniques in order to tackle these issues. Tasks scheduling has been extensively studied in the literature. The problem has been studied with different parameters and objectives. In this article, we address the problem of energy consumption and efficient resource utilization in virtualized cloud data centers. The proposed algorithm is based on task classification and thresholds for efficient scheduling and better resource utilization. In the first phase, workflow tasks are pre-processed to avoid bottlenecks by placing tasks with more dependencies and long execution times in separate queues. In the next step, tasks are classified based on the intensities of the required resources. Finally, Particle Swarm Optimization (PSO) is used to select the best schedules. Experiments were performed to validate the proposed technique. Comparative results obtained on benchmark datasets are presented. The results show the effectiveness of the proposed algorithm over that of the other algorithms to which it was compared in terms of energy consumption, makespan, and load balancing.

Keywords: cloud computing; energy consumption; task scheduling; load balancing; makespan; PSO

check for updates

Citation: Malik, N.; Sardaraz, M.; Tahir, M.; Shah, B.; Ali, G.; Moreira, F. Energy-Efficient Load Balancing Algorithm for Workflow Scheduling in Cloud Data Centers Using Queuing and Thresholds. *Appl. Sci.* **2021**, *11*, 5849. https://doi.org/10.3390/app11135849

Academic Editor: Eui-Nam Huh

Received: 27 May 2021
Accepted: 18 June 2021
Published: 23 June 2021

1. Introduction

Cloud computing provides ubiquitous, convenient, and on-demand services and resources over the internet. A shared pool of configurable computing resources is used to provide these services. Services in cloud computing can be accessed and managed with less effort and fewer interactions [1]. The world has become a global village with the use of the internet through remote access to services and hardware from distant locations. The services that are present over the internet are a revolution in the field of computing in this era. Complex jobs need more and more computational power to execute. Sophisticated and high-performance computing is needed to execute these jobs. Rather than purchasing new hardware, it is a better option to pay per use of services of high-performance computing hardware. Cloud service providers provide such facilities to users so they can access the resources and services by using a pay-per-use model [2]. In the last few years, the number of cloud data centers has increased due to the suitability of storage and computation services for a large number of applications. The services of cloud computing are classified into three main categories, i.e., software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [3]. Users have no geographical restrictions, i.e., they can access services from anywhere at any time [4]. Cloud computing provides virtualized resources for handling various requests for different tasks [5]. The infrastructure of a cloud data center usually consists of thousands of large computing hosts with high-speed computing resources.

A huge amount of data is generated every year; thus, a high processing power and storage capacity are required. Many scientific fields, such as astronomy, bioinformatics, meteorology, environmental science, and geological sciences, deal with large-scale data [6]. The processing of the huge amounts of data generated by these scientific fields severely degrades the performance of clouds [7]. It is a quite challenging task to ensure efficient task scheduling in cloud computing in order to improve the performance of the cloud. In a cloud computing environment, scheduling can be performed at different layers of service, i.e., IaaS, PaaS, and SaaS [8]. Load balancing is used to distribute loads among available resources in a way that the overall load is balanced. The load balancing algorithm receives the requests and distributes user requests among available resources i.e., virtual machines (VMs). The task of the load balancer is to determine the load of available resources and distribute the load among resources. If resources are not utilized with a proper load balancing algorithm, the quality of service (QoS) will be degraded.

In this paper, we present an energy-efficient scheduling algorithm for workflow scheduling in cloud computing. The objective of the proposed algorithm is to reduce the energy consumption, makespan, with berrer load balancing. The proposed algorithm works in two phases, i.e., pre-processing and optimization with Particle Swarm Optimization (PSO). In the first phase, workflow tasks are placed in the respective queues according to the number of levels and length of the tasks. Thresholds are used to allocate resources to tasks to balance the load among resources. In the next phase, PSO is used to optimize scheduling and find better solutions. The paper is organized as follows. Section 2 presents a literature review, followed by the presentation of the materials and methods in Section 3. Section 4 presents the results and a discussion. Finally, Section 5 concludes the article.

2. Literature Review

Workflow scheduling and VM consolidation have been studied extensively in the literature. The scheduling of workflows in cloud environments has become popular due to the extensive applications in both scientific and business areas. Many task scheduling algorithms have been presented in various articles. The solutions consist of heuristic approaches, meta-heuristics, or a combination of both heuristics and meta-heuristics. In this section, we present a review of the different meta-heuristic methods for load balancing and workflow scheduling in cloud computing. There are also review articles with more details on resource allocation and scheduling.

The energy efficiency in cloud data centers has been widely studied. The authors of [9] proposed an energy-efficient algorithm for reducing the energy consumption and improving the resource utilization in cloud data centers. The method is based on the classification of tasks and VMs in order to reduce scheduling time. The scheduling mechanism uses historical task scheduling data to classify user tasks and selects VM types accordingly. The algorithm targets resource utilization, energy consumption, and fault tolerance. The mechanism of merging the same task types minimizes the mean response time and reduces the overall energy consumption. Gill et al. [10] proposed an algorithm called BULLET for scheduling workflows. The algorithm is based on PSO, and it schedules workloads in cloud computing by using QoS metrics according to the user's requirements. QoS requirement metrics are established and the weight of each service is mentioned, such as the consumed energy, execution time, and execution cost. When the user asks for QoS, the services are checked in terms of their QoS metrics and a weight is calculated according to the demands. A workload analyzer analyzes the loads of different workloads and checks whether a workload is feasible for porting to the cloud. When a workload is feasible, it is submitted to the workload manager. The workload manager analyzes the workload characteristics described by the QoS metrics for clustering. The K-Means clustering algorithm is used for clustering. Resource information metrics include information on the available resources, such as the CPU, memory, resource cost, number of resources, and types of resources. The resource provisioner provides the available resources and, finally, the resource scheduler executes the workload on the provisioned resources. Workflow scheduling using hybrid GA-PSO in cloud computing was presented in [11]. GA-PSO starts with random chromosomes to generate the initial population and defines the number of iterations for which to execute the workflow tasks in the cloud. The initial population is processed using the basic operators of the GA, i.e., selection, crossover, and mutation, in order to find the fittest chromosomes. The fittest chromosomes are passed to the PSO algorithm as the initial population. The number of iterations is equally divided between the GA and PSO, i.e., if the number of iterations in GA-PSO is n, then n/2 iterations are processed with GA, and the second half is processed by PSO. In the end, the fittest particles are selected as the solution to the workflow problem. Decreasing the number of initial populations decreases the complexity of the algorithm. Another technique for task scheduling based on a two-way strategy was proposed in [12]. In the initial phase, the algorithm applies a Bayes classifier in order to classify tasks using historical scheduling data. Pre-created VMs are used to process the tasks. In case the classification does not match with the pre-created VMs, the method creates a specified number of VMs according to the classification of the tasks. Pre-created VMs are used to save time during the scheduling phase. In the second phase, task requirements are matched with VMs and scheduled dynamically. The proposed task scheduling method shows more accuracy and consumes less energy compared to the standard methods, i.e., Min-min and Max-min. The authors of [13] proposed a hybrid technique to tackle the problem of workflow scheduling. The problem was considered as a multi-objective optimization considering the cost, makespan, and load balancing as measurement parameters. PSO was used for optimization. In the first phase, pre-processing was used, followed by optimization. The proposed algorithm was validated with comparative experimental results.

The authors of [14] presented a novel solution for VM allocation in cloud data centers. The method was based on the Krill Herd algorithm. The algorithm uses VM aggregation and host shutdown for power management. During this process, QoS is maintained. Efficient integration and the selection of convenient VM migration strategies reduce energy consumption. The results showed that the proposed method reduced energy consumption compared to other algorithms. A scheduling solution based on a non-linear mixed-integer programming model was presented in [15]. The proposed mechanism makes a tradeoff between execution time and energy consumption during the resource allocation phase. A backward technique is applied to adjust task scheduling and achieve the desired goals. The experimental results showed that the proposed approach significantly reduced the computation time for large sizes of parallel applications. Attiqa et al. [16] proposed a Multi-Objective Genetic Algorithm (MOGA) that was used to schedule workflows in cloud environments. The MOGA gave a solution that reduced the makespan and provided an energy-efficient solution in a cloud environment. The results of the proposed algorithm showed a significant enhancement in terms of budget, deadline, and energy consumption. The technique also improved resource utilization. Verma et al. [17] proposed a Hybrid PSO (HPSO) method for minimizing the execution time and execution cost in cloud computing. The algorithm is a hybridization of the Budget and Deadline Constraint Heterogeneous Earliest Finish Time (BDHEFT) and multi-objective PSO. The HPSO optimizes the two contradictory parameters—makespan and cost—under the budget and deadline constraint. In addition to these two parameters, energy consumption is also considered as a parameter. The proposed algorithm gives a set of the best solutions, from which the user can select the best solution. In another study [18], a parallel Genetic Algorithm (GA) was used to solve the multi-objective optimization problem of workflow scheduling. The makespan, cost, and load balancing were the targeted parameters. The proposed algorithm first calculates the best solution for each parameter in parallel, and then the best solution is found for all parameters. The proposed method was evaluated with comparative experimental results. Another algorithm that minimizes energy consumption and Service-Level Agreement Violations (SLAVs) during VM consolidation was proposed in [19]. The authors used an energy-aware VM consolidation algorithm that minimized SLAVs. The algorithm uses a fine-tuned prediction model for VM migration, CPU and memory utilization prediction, and target host selection. The proposed method was evaluated with the Planet Lab datasets. The comparative results showed that the proposed algorithm significantly reduced energy consumption and SLAVs. The authors of [20] presented a technique for VM placement that was based on an improved permutation-based GA and a multidimensional resource-aware best-fit allocation strategy. The objective of the algorithm was to reduce energy consumption by reducing the active hosts. The comparative results showed that the proposed mechanism consumed less energy with fewer active hosts in comparison to other methods. Moreover, the proposed method also yielded better resource utilization and load balancing. Another technique based on prediction and dynamic resource-table-updating mechanisms was proposed in [21]. The objective of the proposed method was energy efficiency in task scheduling. The algorithm considers the tasks' arrival times and the sizes of the tasks for efficient scheduling. The results are computed in terms of the completion time and response time. The simulation results showed the efficiency of the proposed algorithm over the other techniques with which it was compared. The authors of [22] proposed a task scheduling algorithm and workload classification architecture for VM placement in cloud data centers. The objectives of the technique were resource utilization and energy consumption. The proposed algorithm was validated with comparative experimental results. Cloud data centers demand a huge amount of energy, which changes at different hours of operation, together with the utilization of resources. A mechanism for handling the energy consumption in different operating hours was proposed in [23]. The proposed technique uses frequency scaling and non-power-aware hosts to achieve the desired objectives. VM consolidation policies, i.e., the utilization of local regression minimization and static-threshold random selection, were used. The comparative results showed the improvements gained by the proposed method over the other methods with which it was compared. Ensuring the QoS in the presence of energy consumption is a challenge for cloud service providers. To address this issue, the authors of [24] proposed an energy-efficient, QoS-aware algorithm for VM consolidation. The proposed algorithm was based on a Markov-chain-based prediction approach in order to measure the load of the hosts, i.e., to identify over-utilized and underutilized hosts. The linear weighted sum approach was used for VM selection and placement during migration. The algorithm targets QoS and energy consumption.

Techniques for energy efficiency suffer from the problem of performance degradation. To tackle this issue, thresholding is used to achieve a balance between the two parameters. The authors of [25] proposed an energy-efficient algorithm for VM consolidation. The objectives of the method were energy efficiency and throughput. The workload of a host was considered in order to set thresholds for resource utilization and manage VM migrations. The proposed method performed well in terms of the desired parameters in comparison with other standard methods. In [26], the authors proposed a scheduling algorithm based on the Deep Q-network (DQN) technique. The objectives of the method were the optimization of energy consumption and makespan. The tradeoff between the two parameters was used to optimize the problem and achieve the desired objectives. The dynamic behavior of the proposed algorithm according to different workload requirements was proved with experimental results. The results showed the effectiveness of the proposed algorithm in comparison to other methods. The authors of [27] used bio-inspired heuristics for VM consolidation. A more desirable environment for cloud computing was considered on the basis of a larger capacity margin and a higher fitness value for the VMs and hosts, respectively. The comparative experimental results were shown to validate the performance of the proposed algorithm. The authors of [28] proposed an algorithm that overcame the issues of load balancing and load scheduling. The algorithm worked with precision and privacy. The authors used a hybrid approach that could allocate tasks by using the Modified Canopy Fuzzy C-means Algorithm (MCFCMA). The algorithm allocates tasks to respective resources with the help of PSO. In the proposed technique, the load is based on the selected tasks, the cluster requests the tasks with the help of the MCFCMA, and it schedules these tasks for each VM. The feature value is calculated with the help of the PSO algorithm. The proposed technique minimizes the execution time, cost, and load. A scheduling algorithm that uses the Traveling Salesman Problem (TSP) solution strategy was proposed in [29]. The algorithm first converts the problem into an instance of the TSP and then applies the solution to the problem. The method consists of three phases, i.e., clustering, conversion, and assignment. In the first phase, a large problem is divided into small-sized clusters. In the conversion phase, the problems are converted into an instance of the TSP. Finally, the clusters are scheduled to available resources in the assignment phase. The proposed technique was validated with comparative experimental results. Another algorithm for efficient resource utilization and energy consumption was proposed in [30]. The proposed algorithm, i.e., ERES, dynamically allocates resources to workflow tasks according to the input tasks. The algorithm dynamically manages the server load, and live migration of the VMs is used to avoid overloaded and underloaded hosts. Simulation results were presented to validate the effectiveness of the proposed algorithm. FIMPSO is another algorithm for load balancing in a cloud computing environment [31]. The hybridization of Firefly and PSO was used in the proposed algorithm. The Firefly algorithm is used to reduce the search space, whereas an improved, modified PSO is used to select the best responses during the resource allocation process. The resource utilization and response time are the targeted parameters of the proposed algorithm. Comparative experimental results were shown to validate the proposed algorithm.

Energy-efficient scheduling in the presence of other constraints, such as resource utilization and fault tolerance, is a challenging issue. There is a need to develop solutions that address the problem while considering these parameters. This paper presents a scheduling algorithm in order to address the above-mentioned problem. The proposed algorithm is based on task classification, queueing, and thresholds for allotting VMs to tasks and creating the necessary resources. Queuing is used to hold tasks with different intensities in order to reduce resource allocation time. Task classification is used to find suitable VMs in order to reduce the scheduling time. Thresholds are used to efficiently utilize cloud resources and reduce the failure ratio.

3. Materials and Methods

This section presents the proposed algorithm in detail. The algorithm targets the makespan, energy consumption, and load balancing. The algorithm consists of two phases, i.e., preprocessing and PSO-based optimization. First, we discuss the workflow and cloud model, followed by the details of each phase.

3.1. Workflow and Cloud Model

Workflow applications consist of a set of tasks with dependencies, i.e., execution and data dependencies. In the prior case, the tasks have a parent–child relationship. A child task cannot start execution until all parents of that task have completed the execution. In the latter case, the tasks share data, i.e., the output generated by some tasks becomes the input for other tasks. These dependencies make it difficult for a scheduler to effectively schedule resources for workflow applications. Workflow tasks are represented as a directed acyclic graph (DAG), e.g., D (V, E), where E represents edges of the graph and V represents the vertices. Cloud computing consists of virtualized resources, which are referred to as VMs. The goal of a scheduling algorithm is to allocate R_i to W_j, where R_i is the ith resource from a pool of VMs $(VM_1, VM_2, VM_3, \ldots, VM_n)$ and W_j is the workflow application $(W_i, W_2, W_3, \ldots, W_m)$. The goal is to minimize the energy consumption and execution time with a balanced load among resources. The resources have pre-allocated capacities of processing, memory, storage, bandwidth, etc. The processing capacity C_i of a resource VM_i is computed with the number of processing elements (PEs) and the MIPs of each PE, as shown in Equation (1). The resources have pre-allocated capacities of processing, memory, storage, bandwidth, etc. The processing capacity C_i of a resource VM_i is computed with the number of processing elements (PEs) and the MIPs of each PE, as shown in Equation (1).

$$C_i = (PE_{\times} MIPS_i) \tag{1}$$

The capacity of n resources or VMs is calculated with Equation (2).

$$C = \sum_{i=1}^{n} C_i \tag{2}$$

Each VM has a resource utilization at a particular time, which is referred to as the load of the VM. The load is calculated with Equation (3), where TL is the total length of the tasks being processed by VM_i, and C_i is the capacity of VM_i.

$$L_{vmi} = \frac{TL}{C_i} \tag{3}$$

Equation (4) is used to calculate the load L of all VMs.

$$L = \sum_{i=1}^{n} L_{vmi} \tag{4}$$

Load balancing is measured as the load among different nodes in the cloud environment, as shown in Equation (5).

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(L_{vmi} - \bar{L})^2}{n}} \tag{5}$$

where L_{vmi} is the load of VM_i, $\bar{L}$ is the average load of all VMs, and n is the number of VMs.

In cloud computing, the energy consumption is strongly influenced by the utilization of resources. The utilization can be calculated with Equation (6).

$$U = \alpha \frac{\sum_{i=1}^{n} c_i}{C} + \beta \frac{\sum_{i=1}^{n} m_i}{M} \tag{6}$$

where n is the number of VMs running on host h, and c_j, m_j refer to the computing and memory allocated to VM_i. In Equation (6), C and M are the total processing capability and memory of the host, and α and β are the weight parameters of each resource.

The energy consumption can be calculated with Equation (7), where k represents the operational energy consumption, i.e., the idle mode. E_{max} represents the energy consumption during the peak utilization of the processors, and U represents the utilization of the resources of the host calculated with Equation (6).

$$E_c = E_{idle} + (E_{max} - E_{idle}) \times U \tag{7}$$

Workflow tasks may need data for processing. The completion time of the workflow includes both the processing time and the time consumed in obtaining the required data. Equation (8) is used to calculate the completion time of task t_i.

$$Time(t_i) = Time((Trans_{t_i}, t_j) + Time_E(t_i, VM_k)) \tag{8}$$

In Equation (8), $(Trans_{t_i}, t_j)$ is the time consumed by transmitting data from task t_i to t_j, and $Time_E(t_i, VM_k)$ is the execution time of t_i on VM_k. Both parameters are calculated with Equations (9) and (10), respectively.

$$Trans(t_i, t_j) = \frac{sizeof(t_i, t_j)}{\beta(VM_k, VM_m)} \tag{9}$$

In Equation (9), $sizeof(t_i, t_j)$ represents the amount of data that task t_i transfers to task t_j, and $\beta(VM_k, VM_m)$ represents the bandwidth used by VM_k and VM_m. If both VMs are located in the same data center, the transmission cost is neglected.

$$T_E = \frac{l_i}{C_{m_j}} \tag{10}$$

where l_i represents the length of tasks i and C_{m_j} represents the processing capacity of VM_j, which was calculated with Equation (1). The makespan is the total execution of all tasks in a workflow. The makespan can be calculated with Equation (11), where MS refers to the makespan and FT is the finishing time of a task.

$$MS = FT_{i=1}^{n}[task_i time] \tag{11}$$

3.2. Particle Swarm Optimization (PSO)

PSO is an optimization technique that is used to solve multimodal optimization problems. The technique is based on swarms of birds or schools of fishes, where possible solutions are generated as swarms of particles and each position represents a possible solution. Two values, i.e., velocity and position, are used to represent a particle. Initially, random positions are used for all particles. The particles move in the search space to find the best solutions. Two factors, i.e., pbest, which refers to a particle's best position, and gbest, which refers to the global best position, influence the final solution. The position and velocity are updated in each iteration of the execution [8]. Equation (12) shows the function for calculating the velocity [29].

$$v_{id}^{t} = wv_{id}^{t-1} + c1r1(pbest_{id}^{t} - x_{id}^{t}) + c2r2(gbest_{id}^{t} - x_{id}^{t}) \tag{12}$$

In Equation (12), v_{id}^{t} represents the velocity of the dimension d of the particle i in time t. Equation (13) is used to update the position of a particle. In Equation (13), p_{id}^{t} refers to the position of particle i at time t in the d_{th} dimension, and v_{id}^{t} is the velocity calculated in Equation (12) [13].

$$p_{id}^{t+1} = p_{id}^{t} + v_{id}^{t} \tag{13}$$

The fitness of the solution is calculated with the minimum and maximum values of energy consumption, makespan, and load balancing. The weight factor is used for each parameter. In the experiments of this article, an equal weight, i.e., 0.33, was used for all parameters. Equation (14) is used to calculated the fitness value [13].

$$Fitness = w\frac{max^{ec} - ec}{max^{ec} - min^{ec}} + w\frac{max^{lb} - lb}{max^{lb} - min^{lb}} + w\frac{max^{ms} - ms}{max^{ms} - min^{ms}} \tag{14}$$

The scheduling problem can be formulated as a multi-objective optimization problem. The problem is represented with m decision vectors $x = (x_1, \ldots, x_m) \in X$ in search space X and n objectives $y = (y_1, \ldots, y_n) \in Y$ in objective space Y, as shown in Equation (15) [13].

$$min(y = f(x) = [f_1(x), \ldots, f_n(x)]) \tag{15}$$

As the problem is considers multiple objectives, it is difficult to claim an optimal solution for all objectives. Usually, the Pareto optimal set is taken from many solutions. The Pareto optimal set is a solution that is considered optimal for a set of objectives. For two decision vectors x^1 and x^2, the dominance of decision vector x^1 over x^2 is defined as shown in Equation (16). The equation shows that the decision vector x^1 is as good as x^2 for all objectives, and x^1 is strictly superior to x^2 in at least one objective [13].

$$x^1 \prec x^2 \iff \forall i f_i(x^1) \le f_i(x^2) \wedge \exists j f_j(x^1) < f_j(x^2) \tag{16}$$

If none of the decision vectors in the solution space dominate the decision vector x^1, then x^1 is said to be Pareto optimal, as shown in Equation (17) [13].

$$\nexists x^2 \in X : x^2 \prec x^1 \tag{17}$$

A Pareto optimal set consists of all Pareto optimal decision vectors, as shown in Equation (18), and the Pareto optimal front refers to the image of the Pareto optimal set, as shown in Equation (19) [13].

$$P_S = \{x^1 \in X, | \nexists x^2 \in X : x^2 \prec x^1\} \tag{18}$$

$$P_F = \{f(x) = (f_1(x), \ldots, f_n(x)) | x \in P_S\} \tag{19}$$

3.3. Proposed Algorithm

The proposed algorithm targets the dependencies between tasks and the execution times of tasks in order to reduce the makespan and energy consumption and to balance the loads among resources. VMs are allocated to tasks according to the demands of the workflow tasks to ensure a balanced load among the resources. The proposed algorithm reads the workflow tasks and sets thresholds for the depth and length of the tasks, where depth refers to the levels of tasks and length is the execution time of the tasks. The threshold values are used to process tasks according to different priorities during their execution. Tasks with more dependencies create bottlenecks in the system and cause long execution times. Similarly, tasks with longer execution times need to be processed with priority to reduce the overall execution time. These tasks are processed with high priority, i.e., VMs with high processing capabilities are allocated to these tasks. The proposed algorithm also searches for tasks with long execution times and processes these tasks with priority in order to avoid unnecessary waiting of tasks at the same level. Thresholds are defined for both the number of dependencies and the time according to the input data. These steps are used to reduce the execution time, which also leads to reduced energy consumption as resources are utilized efficiently. Algorithm 1 shows the steps involved in the procedure.

Algorithm 1: Avoid bottleneck tasks

```
Input   : workflow w
Output: Queues of tasks based on depth and length
Assign thresholds dt for depth of tasks and lt for length of tasks
for each task t in the task list do
    depth = number of levels dependent on t
    length = execution time of t
    if depth>= dt then
        move t to depth queue
    end
    if length>= lt then
        move t to length queue
    end
end
Return queues
```

In the next phase, the proposed algorithm uses queues according to the intensities of tasks, i.e., for CPU-intensive tasks and tasks with more dependencies, separate queues are maintained. Putting tasks with different intensities in separate queues saves time in finding the suitable VMs during the VM allocation process. In addition to the intensities of tasks, other information related to tasks, i.e., arrival time and the deadline, is used to store tasks in queues. After classifying tasks into different queues, the next step is to create suitable VMs for the tasks. For this purpose, historical scheduling records (HSRs) are used. If there is no matching record in the HSRs, a new VM is created with the necessary resources needed to process the task, and the HSRs are updated accordingly. The steps involved in the proposed algorithm are shown in Algorithms 2–4.

Algorithm 2: Create VM types

```
Input  : Historical scheduling record (HSRs), task lists (from specified queues),
Output : VMs (Types)
N = number of tasks in a queue
for (each task t in N) do
    compute P($T_t$) (Equation 21)
    H = best n P($T_t$)
end
for (each l in H) do
    if (t found in HSRs) then
        allocate $VM_j$ based on type of t
        else
            CreateVM()
        end
        Assign $VM_j$ to task t
    end
end
```

Algorithm 3: Create VMs

```
Input  : List of tasks $L_t$ from queues from algorithm 1 , list of hosts $L_h$
Output : List of VMs
for (each host h in $L_h$) do
    u = resource utilization of h
    if (host resources are available) then
        Create VM
        Update HSRs
    end
    else
        Power on new host
        Create VM
        Update HSR
    end
end
```

Algorithm 4: Task Scheduling

```
Input  : List of tasks $L_t$ from queues from algorithm 1, list of VMs $V_l$
Output : Schedule of tasks for VMs
for (each task t in $L_t$) do
    Classify tasks as CPU intensive, memory intensive, or both CPU and memory
     intensive
    Store each category in separate queues
    $V_l$=VMtypes()
    for (each VM v in $V_l$) do
        calculate the matching degree of v and t
        if (the desired condition is satisfied) then
            schedule t to v
        else
            $V_n$=CreateVM()
            schedule t to $V_n$
            update HSRs
        end
    end
end
```

In the first phase, the tasks are marked, i.e., suitable VMs are found for the tasks. The types of tasks are classified accordingly, and the VM types are determined. For a set of tasks T extracted from the historical data, let T_l represent a type l task in T. The ratio P can be calculated with Equation (20) [6,11].

$$P = \frac{|T_l|}{|T|} \tag{20}$$

Let task T_i^r represent a candidate task, where r = {1,2,3,4} represents the CPU, memory, bandwidth, and storage requirements of the task and V_j^r represents a VM in which r = {1,2,3,4} represents the CPU, memory, bandwidth, and storage capacity of the VM, respectively. The matching degree of task T_i^r with VM V_j^r can be calculated with Equation (21) [6,11].

$$P(T_i^r|V_j^r) = \begin{cases} (V_j^r/T_i^r)^2, & \text{if } T_i^r > V_j^r. \\ (V_{max}^r - V_j^r + T_i^r)/V_{max}^r, & \text{otherwise.} \end{cases} \tag{21}$$

where $V_{max}^r = k \in U^{max} V_k^r$ and k represents the type of VM. The possibility that a task T_j belongs to type Y_j can be calculated with Equation (22) [11].

$$P(Y_j|T_i) = \Pi_{r=1}^4 P(T_i^r|V_j^r) \tag{22}$$

The proposed algorithm uses PSO to optimize the scheduling problem. After the preprocessing phase, scheduling with PSO starts. Initially, particles are positioned randomly with a random velocity. During the execution, the particles are updated with the desired fitness function. The process is repeated iteratively, and the variables are updated in each iteration. The proposed algorithm uses the fitness function shown in Equation (15). Algorithm 5 shows the steps. The pseudocode of Algorithm 3 is a modified version of that from [13].

The mapping of resources to tasks in the presence of multiple objectives is a complex and challenging task [28]. In this scenario, a search space is created according to the number of tasks in the workflow. The dimensions of the search space are set to be equal to the number of tasks. The values of the dimensions are taken according to the number of VMs, i.e., from 1 to the number of VMs. In this study, the notations from previous studies [13,32] are used to represent the mapping of tasks to VMs, i.e., ${x_i}^t = (x_{i1}^t, x_{i2}^t, \dots, x_{ij}^t)$, where x_{ij}^t represents that, at time t, the jth place of a particle is assigned to VM_i. The number of tasks in the workflow represents the dimensions of the search space. The velocity is represented by ${v_i}^t = (v_{i1}^t, v_{i2}^t, \dots, v_{ij}^t)$, where v_{ij}^t represents the velocity, which shows that, at time t, VM_i moves to the jth place of a particle with velocity v [13]. In successive iterations, the algorithm finds non-dominated solutions. These solutions are stored in the archive. These solutions are called feasible solutions. Initially, the archive is empty and solutions are stored as long as the algorithms find non-dominated solutions. New solutions are added to the archive only if the new solutions dominate the current solutions. The dominance of the solutions is calculated with the fitness function used. Finally, the archive only contains feasible solutions, which are also referred to as non-dominated solutions.

Algorithm 5: PSO-based scheduling algorithm

```
Input  : List of tasks, VM list
Output : Mapping of tasks to VMs
$P$ = initial population
$p$ = $i_{th}$ particle from $P$
Evaluate $p$
Calculate velocity of $p$
$gbest$= global best position
$pbest$ = particles' best position
for each particle $p$ in $P$ do
    for each task $t$ in workflow $w$ do
        initialize $X_{ij}^t$ randomly
        initialize velocity $v$ randomly
        evaluate $p_i$
        update $pbest$ and $gbest$
    end
end
Return the mapping of tasks to VMs
```

4. Results and Discussion

This section presents the experimental evaluation of the proposed algorithm. Benchmark workflows were used to validate the proposed algorithm [33]. The datasets consist of different workflows with varying numbers of tasks and varying dependency levels. The datasets have been used to validate many scheduling algorithms in cloud environments. Table 1 and Figure 1 show the details and structure of the workflows, respectively [13].

Standard algorithms, i.e., GA and PSO, were used in the comparison as a baseline. Specialized schedulers [30,31] were also included in the comparison. The algorithms were selected based on their relevance to the problem. The experiments were carried out on an Intel Core i3 processor equipped with 8 GB of memory running on the Ubuntu 16.04 operating system. The algorithms were evaluated in terms of makespan, energy consumption, and balancing degree. CloudSim [34] was used to simulate the algorithms. The simulation was performed with 10 VMs and three data centers. VMs with different specifications were selected for simulation. Each VM was allocated 1000 MBs of memory and MIPS from 1500 to 3000. Processing, memory, storage, and transfer costs were set to

0.017, 0.05, 0.01, and 0.01, respectively. The GA was evaluated with a crossover probability of 0.8 and a mutation probability of 0.2. PSO was executed with the value of inertia weight factor ω of 1.2, and the learning factors were set to 2. The other algorithms were executed with the parameters specified in the respective articles. Each experiment was repeated 20 times, and the average values were selected for comparison.

Table 2 shows the results of the proposed algorithm compared to those of the other algorithms. The proposed algorithm yielded better results for all parameters on all datasets. In comparison to the standard algorithms, i.e., PSO and GA, the proposed algorithm performed better and at higher rates than the specialized algorithms. In the specialized schedulers, ERES remained closest to the proposed algorithm in terms of energy consumption. In terms of makespan and load balance, FIMPSO remained close to the proposed algorithm on the majority of the datasets. For some datasets, the energy consumption of FIMPSO was better than that of ERES and was close to the proposed algorithm.

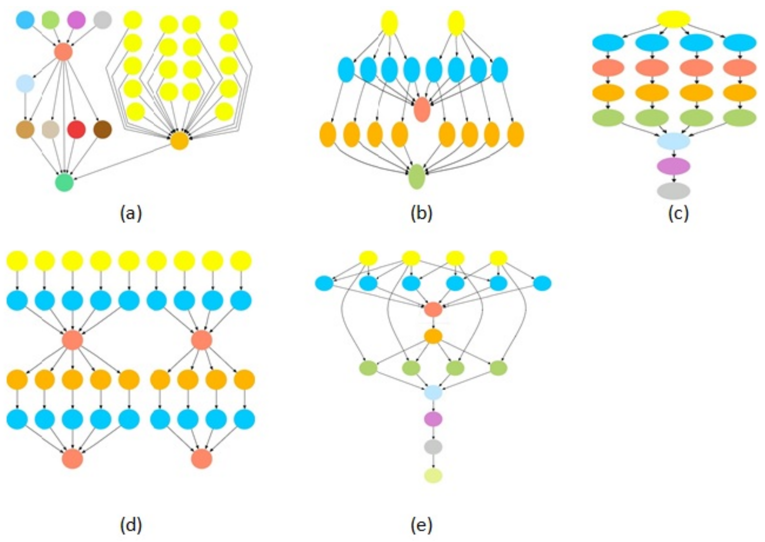

Figure 1. Structures of the workflows used for the experiments: (**a**) Sipht, (**b**) CyberShake, (**c**) Epigenomics, (**d**) LIGO, and (**e**) Montage.

Table 1. Datasets used for the experiments.

Dataset	Nodes	w-Levels	Parallel Tasks	Average File Size (MBs)	Average Execution Time (s)
Montage	100	9	62	18.05	10.19
Sipht	100	7	51	22.34	173.34
LIGO	100	8	24	28.8	209.78
CyberShake	100	5	48	999.41	21.82
Epigenomics	100	8	24	4033.59	1277.21

Figures 2–6 show the percent improvement gain of the proposed algorithm over the other algorithms. For the Montage dataset, in terms of makespan, the proposed algorithm achieved percent improvement gain of 32.60, 31.36, 22.8, and 21.18 over PSO, GA, ERES, and FIMPSO, respectively. For the same dataset, the improvement gains in terms of energy consumption over the compared methods were 25.1, 24.01, 18.35, and 20.99. In terms of load balance, the percent improvement gains of the proposed algorithm over PSO, GA, ERES, and FIMPSO were 31.55, 32.75, 24.13, and 22.22, respectively. The results are shown in Figure 2.

Table 2. Results of the proposed algorithm compared with those of standard and specialized schedulers on different workflows. Energy consumption is represented in KWh.

Parameters	**PSO**	**GA**	**ERES**	**FIMPSO**	**Proposed**
			Montage		
Makespan	276	271	241	236	186
Energy Consumption	232.31	229	213.12	220.25	174
Load Balance	225	229	203	198	154
			Sipht		
Makespan	5136	4987	4536.23	4592	3985
Energy Consumption	886	882	806.29	802	713
Load Balance	204	202	184	182	159
			LIGO		
Makespan	4852	4782	4232.36	4173	3374
Energy Consumption	1052	1023	966	936	788
Load Balance	236	227	202	198	155
			CyberShake		
Makespan	6431	6386	5631.72	5451	4523
Energy Consumption	188	185	161	157	134
Load Balance	246	241	218	214	176
			Epigenomics		
Makespan	70,836	70,332	40,257.17	39,788	32,902
Energy Consumption	15,732	15,655	13,849	13,725	10,736
Load Balance	253	248	227	223	176

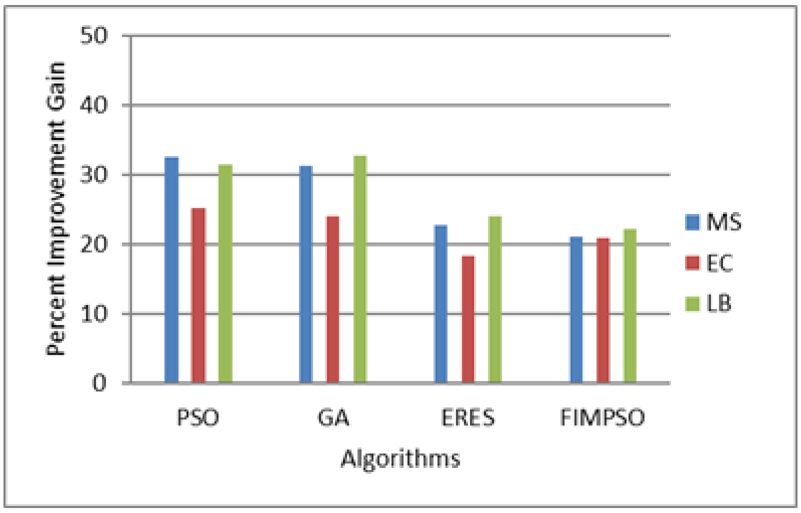

Figure 2. Percent improvement gain of the proposed algorithm over the other methods on the Montage dataset.

In the case of the Sipht dataset, the percent improvement gains in terms of makespan over PSO, GA, ERES, and FIMPSO were 22.41, 20.09, 12.15, and 13.21, respectively. In terms of energy consumption, the improvement gains of the proposed algorithm were 19.51, 19.14, 11.55, and 11.10 over PSO, GA, ERES, and FIMPSO. In terms of load balance, the proposed algorithm achieved percent improvement gains of 22.05, 21.28, 13.58, and 12.63 over the other algorithms. Figure 3 shows the results of the percent improvement gains of the proposed algorithm over the other algorithms.

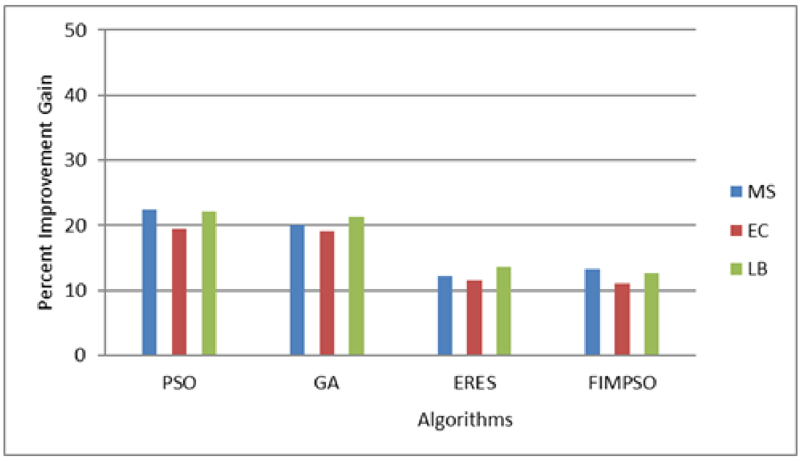

Figure 3. Percent improvement gain of the proposed algorithm over the other methods on the Sipht dataset.

The proposed algorithm also achieved similar improvements compared to the other methods on the LIGO dataset. Figure 4 shows the results for the LIGO dataset. The results show that the proposed algorithm achieved percent improvement gains of 30.46, 29.44, 20.28, and 19.14 in terms of makespan over the other methods. For the same dataset, the improvement gains in terms of energy consumption over the compared methods were 25.09, 22.97, 18.45, and 15.81. In terms of load balance, the improvement gains of the proposed algorithm over PSO, GA, ERES, and FIMPO were 34.32, 31.71, 32.26, and 21.71, respectively.

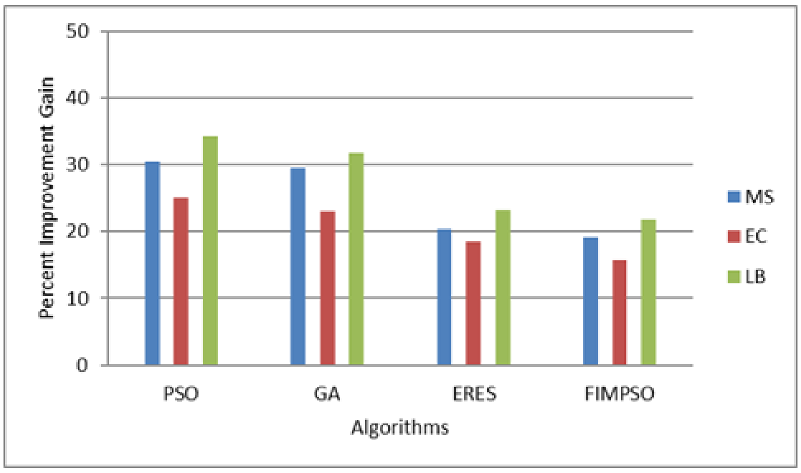

Figure 4. Percent improvement gain of the proposed algorithm over the other methods on the LIGO dataset.

Figure 5 shows the results of the percent improvement gain of the proposed algorithm over the other algorithms for the CyberShake dataset. The improvement gains in terms of the makespan over the other methods for this dataset were 29.66, 29.17, 19.68, and 17.02. The improvement gains over PSO, GA, ERES, and FIMPSO in terms of energy consumption were 28.72, 27.56, 16.77, and 14.64, respectively. For the same dataset, the improvement gains in terms of load balance were 28.45, 26.97, 19.26, and 17.75 over PSO, GA, ERES, and FIMPSO, respectively.

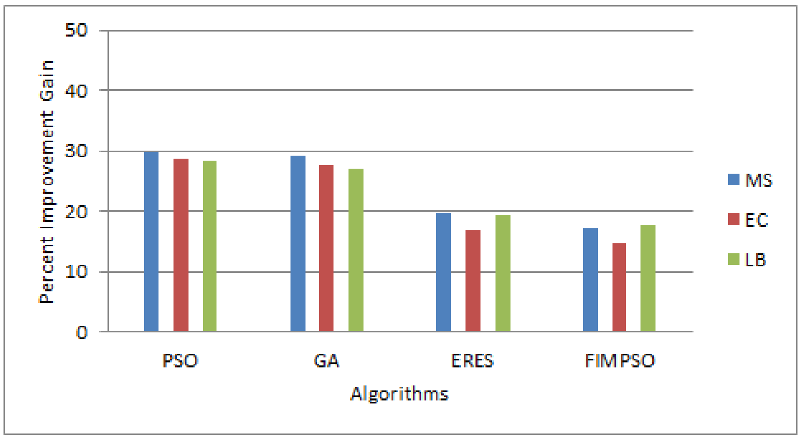

Figure 5. Percent improvement gain of the proposed algorithm over the other methods on the Cybershake dataset.

In the case of the Epigenomics datasets, the proposed algorithm achieved percent improvement gains of 53.55, 53.21, 18.27, and 17.30 over the other algorithms. In terms of the energy consumption, the proposed algorithm achieved percent improvement gains of 28.72, 27.56, 16.77, and 14.64 over PSO, GA, ERES, and FIMPSO, respectively. For the same dataset, the improvement gains of the proposed algorithm over the other methods in terms of load balance were 28.45, 26.97, 19.26, and 17.75. Figure 6 shows the detailed results.

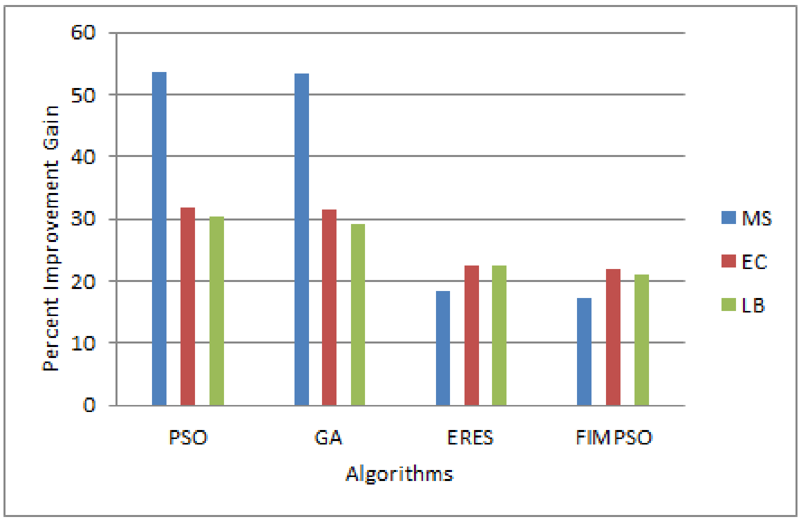

Figure 6. Percent improvement gain of the proposed algorithm over the other methods on the Epigenomics dataset.

5. Conclusions

Cloud computing is a technology that is widely used in many domains, including scientific applications. Due to the large numbers of users and applications, cloud computing environments suffer from some issues. These issues include energy consumption, fault tolerance, user deadlines, etc. Scheduling and VM placement are techniques that are used to handle these issues. There are many solutions for scheduling cloud resources. This article addresses the problem of energy efficiency and resource utilization in cloud data centers. The proposed work is based on task classification, thresholds, and queueing. In the first phase, workflow tasks are placed in queues according to the number of levels and the execution times of the tasks. The tasks are classified accordingly, and resources are created. The proposed algorithm was validated on benchmark datasets, and comparative experimental results were presented in terms of the makespan, energy efficiency, and load balancing. The results were compared with those of standard algorithms and specialized schedulers. The results showed that the proposed algorithm achieved better results than those of the other methods in terms of all parameters. A percent improvement gain from 13 to 53 percent was achieved.

The issues of VM migration and adaptive thresholds require further investigation in order to further improve the solution and achieve better results. The scheduling problem consists of many parameters, but some parameters are contradictory. The designed solutions must consider effects on other parameters while improving the desired parameters. Security and privacy are examples of such parameters that need to be addressed, in addition to other parameters. The implementation of the simulated algorithm in actual environments will also bring challenges, such as the administration cost, energy consumed by factors other than computing, failures in hardware, and data backups.

Author Contributions: Conceptualization, N.M. and M.S.; methodology, N.M.; software, N.M.; validation, M.T., G.A. and M.S.; formal analysis, M.S. and G.A.; investigation, N.M.; resources, M.S. and M.T.; data curation, N.M., M.T., and G.A.; writing—original draft preparation, N.M.; writing—review and editing, M.S., G.A., B.S. and F.M.; visualization, M.T.; supervision, M.S.; project administration, F.M. and B.S.; funding acquisition, F.M. and B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the FCT—Fundação para a Ciência e a Tecnologia, I.P. [Project UIDB/05105/2020] and Cluster Research Grant R20143 of Zayed University, UAE.

Data Availability Statement: The datasets used are publicly available.

Conflicts of Interest: There is no conflict of interest to report.

References

1. Josep, A.D.; Katz, R.; Konwinski, A.; Gunho, L.; Patterson, D.; Rabkin, A. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58.
2. Muteeh, A.; Sardaraz, M.; Tahir, M. MrLBA: Multi-resource load balancing algorithm for cloud computing using ant colony optimization. *Cluster Comput.* **2021**. [CrossRef]
3. Maryam, K.; Sardaraz, M.; Tahir, M. Evolutionary algorithms in cloud computing from the perspective of energy consumption: A review. In Proceedings of the IEEE 2018 14th International Conference on Emerging Technologies (ICET), Islamabad, Pakistan, 21–22 November 2018; pp. 1–6.
4. Kumar, G.G.; Vivekanandan, P. Energy efficient scheduling for cloud data centers using heuristic based migration. *Clust. Comput.* **2019**, *22*, 14073–14080. [CrossRef]
5. Zhan, Z.H.; Liu, X.F.; Gong, Y.J.; Zhang, J.; Chung, H.S.H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–33. [CrossRef]
6. Qin, X.; Jiang, H. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. *Parallel Comput.* **2006**, *32*, 331–356. [CrossRef]
7. Marahatta, A.; Wang, Y.; Zhang, F.; Sangaiah, A.K.; Tyagi, S.K.S.; Liu, Z. Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers. *Mob. Netw. Appl.* **2019**, *24*, 1063–1077. [CrossRef]
8. Masdari, M.; Salehi, F.; Jalali, M.; Bidaki, M. A survey of PSO-based scheduling algorithms in cloud computing. *J. Netw. Syst. Manag.* **2017**, *25*, 122–158. [CrossRef]
9. Marahatta, A.; Pirbhulal, S.; Zhang, F.; Parizi, R.M.; Choo, K.K.R.; Liu, Z. Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data center. *IEEE Trans. Cloud Comput.* **2019**. [CrossRef]
10. Gill, S.S.; Buyya, R.; Chana, I.; Singh, M.; Abraham, A. BULLET: Particle swarm optimization based scheduling technique for provisioned cloud resources. *J. Netw. Syst. Manag.* **2018**, *26*, 361–400. [CrossRef]
11. Manasrah, A.M.; Ba Ali, H. Workflow scheduling using hybrid ga-pso algorithm in cloud computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1934784. [CrossRef]
12. Zhang, P.; Zhou, M. Dynamic cloud task scheduling based on a two-stage strategy. *IEEE Trans. Autom. Sci. Eng.* **2017**, *15*, 772–783. [CrossRef]
13. Sardaraz, M.; Tahir, M. A hybrid algorithm for scheduling scientific workflows in cloud computing. *IEEE Access* **2019**, *7*, 186137–186146. [CrossRef]
14. Soltanshahi, M.; Asemi, R.; Shafiei, N. Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers. *Heliyon* **2019**, *5*, e02066. [CrossRef]
15. Hu, B.; Cao, Z.; Zhou, M. Scheduling real-time parallel applications in cloud to minimize energy consumption. *IEEE Trans. Cloud Comput.* **2019**. [CrossRef]

16. Rehman, A.; Hussain, S.S.; ur Rehman, Z.; Zia, S.; Shamshirband, S. Multi-objective approach of energy efficient workflow scheduling in cloud environments. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4949. [CrossRef]
17. Verma, A.; Kaushal, S. A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling. *Parallel Comput.* **2017**, *62*, 1–19. [CrossRef]
18. Sardaraz, M.; Tahir, M. A parallel multi-objective genetic algorithm for scheduling scientific workflows in cloud computing. *Int. J. Distrib. Sens. Netw.* **2020**, *16*, 1550147720949142. [CrossRef]
19. Moghaddam, S.M.; O'Sullivan, M.; Walker, C.; Piraghaj, S.F.; Unsworth, C.P. Embedding individualized machine learning prediction models for energy efficient VM consolidation within Cloud data centers. *Future Gener. Comput. Syst.* **2020**, *106*, 221–233. [CrossRef]
20. Abohamama, A.; Hamouda, E. A hybrid energy–Aware virtual machine placement algorithm for cloud environments. *Expert Syst. Appl.* **2020**, *150*, 113306. [CrossRef]
21. Praveenchandar, J.; Tamilarasi, A. Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *12*, 4147–4159. [CrossRef]
22. Kaur, K.; Garg, S.; Aujla, G.S.; Kumar, N.; Zomaya, A. A multi-objective optimization scheme for job scheduling in sustainable cloud data centers. *IEEE Trans. Cloud Comput.* **2019**. [CrossRef]
23. Singh, B.P.; Kumar, S.A.; Gao, X.Z.; Kohli, M.; Katiyar, S. A study on energy consumption of DVFS and Simple VM consolidation policies in cloud computing data centers using CloudSim Toolkit. *Wirel. Pers. Commun.* **2020**, *112*, 729–741. [CrossRef]
24. Tarafdar, A.; Debnath, M.; Khatua, S.; Das, R.K. Energy and quality of service-aware virtual machine consolidation in a cloud data center. *J. Supercomput.* **2020**, *76*, 9095–9126. [CrossRef]
25. Saadi, Y.; El Kafhali, S. Energy-efficient strategy for virtual machine consolidation in cloud environment. *Soft Comput.* **2020**, *24*, 14845–14859. [CrossRef]
26. Peng, Z.; Lin, J.; Cui, D.; Li, Q.; He, J. A multi-objective trade-off framework for cloud resource scheduling based on the deep Q-network algorithm. *Cluster Comput.* **2020**, *23*, 2753–2767. [CrossRef]
27. Wang, J.V.; Ganganath, N.; Cheng, C.T.; Chi, K.T. Bio-inspired heuristics for vm consolidation in cloud data centers. *IEEE Syst. J.* **2019**, *14*, 152–163. [CrossRef]
28. Nanjappan, M.; Albert, P. Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment. *Concurr. Comput. Pract. Exp.* **2019**, e5517. [CrossRef]
29. Nasr, A.A.; El-Bahnasawy, N.A.; Attiya, G.; El-Sayed, A. Using the TSP solution strategy for cloudlet scheduling in cloud computing. *J. Netw. Syst. Manag.* **2019**, *27*, 366–387. [CrossRef]
30. Garg, N.; Singh, D.; Goraya, M.S. Energy and resource efficient workflow scheduling in a virtualized cloud environment. *Cluster Comput.* **2020**, *24*, 767–797. [CrossRef]
31. Devaraj, A.F.S.; Elhoseny, M.; Dhanasekaran, S.; Lydia, E.L.; Shankar, K. Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments. *J. Parallel Distrib. Comput.* **2020**, *142*, 36–45. [CrossRef]
32. Jianfang, C.; Junjie, C.; Qingshan, Z. An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm. *Cybern. Inf. Technol.* **2014**, *14*, 25–39. [CrossRef]
33. Bharathi, S.; Chervenak, A.; Deelman, E.; Mehta, G.; Su, M.H.; Vahi, K. Characterization of scientific workflows. In Proceedings of the IEEE 2008 3rd Workshop on Workflows in Support of Large-Scale Science, Austin, TX, USA, 17 November 2008; pp. 1–10.
34. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]