Energy-Aware Scheduling of Workflow Using a Heuristic Method on Green Cloud

Zhihao Peng,1 Behnam Barzegar,2 Maryam Yarahmadi,3 Homayun Motameni,4 and Poria Pirouzmand1

1School of Computer and Software, Dalian Neusoft University of Information, Dalian 116023, China
2Department of Computer Engineering, Nowshahr Branch, Islamic Azad University, Nowshahr, Iran
3Department of Computer Engineering, University College of Rouzbahan, Sari, Iran
4Department of Computer Engineering, Sari Branch, Islamic Azad University, Sari, Iran

Correspondence should be addressed to Behnam Barzegar; barzegar.behnam@yahoo.com

Received 4 July 2020; Revised 25 August 2020; Accepted 5 September 2020; Published 17 September 2020

Academic Editor: Cristian Mateos

Copyright © 2020 Zhihao Peng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Energy consumption has been one of the main concerns to support the rapid growth of cloud data centers, as it not only increases the cost of electricity to service providers but also plays an important role in increasing greenhouse gas emissions and thus environmental pollution, and has a negative impact on system reliability and availability. As a result, energy consumption and efficiency metrics have become a vital issue for parallel scheduling applications based on tasks performed at cloud data centers. In this paper, we present a time and energy-aware two-phase scheduling algorithm called best heuristic scheduling (BHS) for directed acyclic graph (DAG) scheduling on cloud data center processors. In the first phase, the algorithm allocates resources to tasks by sorting, based on four heuristic methods and a grasshopper algorithm. It then selects the most appropriate method to perform each task, based on the importance factor determined by the end-user or service provider to achieve a solution designed at the right time. In the second phase, BHS minimizes the makespan and energy consumption according to the importance factor determined by the end-user or service provider and taking into account the start time, setup time, end time, and energy profile of virtual machines. Finally, a test dataset is developed to evaluate the proposed BHS algorithm compared to the multiheuristic resource allocationalgorithm (MHRA). The results show that the proposed algorithm facilitates 19.71% more energy storage than the MHRA algorithm. Furthermore, the makespan is reduced by 56.12% in heterogeneous environments.

1. Introduction

With the rapid increase in demand for service-oriented computing, in association with the growth of cloud computing technologies, large-scale virtualized data centers have been established worldwide. These huge data centers consume power at a large scale, which results in a high operational cost [1]. Furthermore, they emit greenhouse gases such as carbon dioxide [2–7] and thus produce adverse effects on the environment [8–10]. The destructive consequences of high energy consumption have attracted the attention of researchers, which has led to the emergence of a new research field called green computing [11–13]. The main idea of green computing is to enable the effective and efficient use of resources, by designing algorithms and methods that can reduce energy consumption. To achieve this goal, data centers must manage their resources with the use of efficient energy reduction techniques [14]. One way to decrease energy consumption is to utilize scheduling policies that allocate tasks to specific resources, which affect processing time and energy consumption [15]. Scheduling is a fundamental solution to improve the productivity of all cloud-based services [16]. Therefore, the issue of scheduling parallel programs with limited priority or a directed acyclic graph (DAG), which is one of the applied models used in scientific and engineering disciplines, on homogeneous and heterogeneous computing systems such as
Cloud data centers with regard to the amount of energy consumed and other performance parameters is of fundamental importance and is considered as one of the problematic optimization issues that many innovative algorithms have been proposed to solve them [2]. A trade-off must be made between the parameters of efficiency and energy consumption, which leads to a lowered makespan and energy consumption and the increase in efficiency reaching the service agreement level. Several methods are available to reduce the power consumption of an application when it is running on a distributed platform. For example, Miyoshi et al. [17] used low-power processor architectures or dynamic frequency and voltage scaling. Tiwari et al. [18] used algorithmic design and effective energy patterns in compilers, and Beloglazoy et al. [19] applied the change in scheduling policies for task-based applications on existing resources. The main focus of most traditional algorithms for scheduling DAG tasks on distributed computing platforms, such as clusters, grades, and cloud computing, has been to minimize makespan while eradicating concerns about the energy consumed by data centers [16–21]. However, there is a trade-off between energy consumption and makespan, as increasing the efficiency to enable faster implementation occasionally requiring higher energy consumption [22, 23]. In turn, high energy consumption increases costs and thus reduces the profitability and efficiency of cloud service providers [8, 9]. Meanwhile, end-users and cloud service providers want their services to be provided at a lower cost and time [24–26]. Cloud end-users generally anticipate completing their tasks without any delay, whereas cloud providers seek to reduce the energy cost, which constitutes as one of the major costs in the cloud service environment. However, lowering energy consumption increases the makespan and leads to customer dissatisfaction [27]. Most existing techniques for minimizing cost and time are designed for traditional computing platforms, which cannot be applied to cloud computing platforms with resource management methods [13]. In this paper, we attempt to minimize makespan and energy consumption based on the importance factor determined by the end-user and the provider, by the utilization of resource allocation in a queue created by four heuristics methods and the grasshopper algorithm. The most appropriate method to implement each task is then chosen using the roulette wheel.

To solve the above problems, this paper makes five main contributions:

(i) Energy-aware run-time scheduler for scheduling of workflow.

(ii) Provides a methodology to automatically generate the required power consumption profile.

(iii) Selects the best sorting method to perform any task between a heuristic method and grasshopper algorithm to increase productivity.

(iv) Offers a model to estimate the energy consumption and makespan of the task.

(v) Evaluates the trade-off between energy saving and makespan for different scenarios.

The article is structured as follows: Section 2 provides an overview of related work, and Section 3 states the problem. Section 4 describes the proposed method. Section 5 expresses the evaluation of the proposed method. The time complexity analysis for the proposed algorithm and MHRA algorithm is presented in Section 6, and finally, Section 7, discusses the conclusion and future work.

2. Related Works

Today, cloud computing plays an essential role in academia and industry [28]. However, due to high customer demand and limited resources, it is necessary to transfer a small amount of work to data centers [29]. This process helps to complete the applications provided to data centers through more flexible and inexpensive use of resources. Because there are so many tasks on cloud servers and resources in the cloud are heterogeneous [30], scheduling tasks in a cloud environment is a consistent challenge. In this section, we briefly describe some existing research in the field of scheduling on the topic of allocating resources to tasks in cloud environments. Barzegar et al. [2] introduced a time- and energy-aware scheduling algorithm called EATSDCD. The paper presents a method that uses a combination of two clustering and duplication strategies and the use of slack time, to schedule DAGs on data center processors with dynamic voltage and frequency scaling capability. In the first phase, by intelligently combining the two strategies of duplication and clustering, the focus is placed on reducing the makespan and energy consumed by processors in an effort to execute DAG while satisfying the throughput constraint. Then, the main idea of the algorithm is to reduce the maximum energy consumption in the second phase. After calculating the slack time for each task and then determining the critical path, the slack time is distributed between the set of noncritical dependent tasks in a cluster, and the frequency of processors with dynamic scale scalability then performs the dependent task set. Then, the frequency of DVFS-enabled processors is scaled down to execute noncritical tasks as well as idle and communication phases, without increasing the makespan of the tasks. Juarez et al. [22] introduced an energy-aware timing algorithm for task-based applications called the multiheuristic resource allocation algorithm (MHRA). The objective of this scheduling algorithm is to minimize the function of the target that combines energy consumption and makespan. These criteria are combined with an importance factor determined end-users and service providers to define which is more important for their objectives: energy saving or makespan. The authors also proposed a model for estimating the energy consumption required to execute efficiently applications on set of resources. Safari and Khorsand [23] investigated the energy-aware schedule algorithm for a workflow with limited time intervals in a homogeneous cloud environment. They were able to reduce makespan, particularly when the number of tasks was large. This offers also a promising way to reduce
energy consumption that takes into account time limits and service quality as a service level contract. Sofia and Ganesh Kumar [27] proposed a multiobjective algorithm based on nondominated sorting genetic algorithm (NSGA-II). Minimizing the energy consumption and makespan from the cloud service, the main objective was NSGA-II. In order to control the energy consumption and efficiency, the DVFS system was incorporated into the optimization procedure, and a set of nondomination solutions are obtained and used. Furthermore, the artificial neural network (ANN), which is a successful machine learning algorithm, was used based on the features of the resources available in it. The authors proved that the number of generations was critically reduced by using ANN in the optimization procedure. Additionally, the computational cost was minimized by optimally choosing the control parameters of genetic algorithm (GA), such as the number of generations, population size, crossover, and mutation probability. Yang et al. [31] proposed an energy-aware importance ratio-based stochastic task scheduling (EISTs) algorithm, which maintains a good balance between the optimization of makespan and energy conservation. The authors claimed that the algorithm achieves a shorter makespan when the importance ratio of makespan to energy consumption is high and a lower energy consumption when the ratio is low. Voorneveld [32] proposed an algorithm based on the Pareto solution that is similar to our method of using the objective function. However, their proposed combination is that of energy consumption and task makespan. In contrast, the method proposed in this paper considers makespan and energy consumption for various elements such as cores, virtual machines, and nodes. Ben Alla et al. [33] proposed efficient energy-aware task scheduling with deadline-constrained in cloud computing (EATS-D). The main objective of their proposed solution is to reduce the energy consumption of the cloud resources, consider different end-user priorities, and optimize the makespan within the deadline constraints. They achieved good performance by minimizing the makespan, reducing energy consumption, and improving resource utilization while meeting deadline constraints. Peng et al. [34] proposed an online resource scheduling framework based on the Deep Q-network (DQN) algorithm. The framework was able to realize a trade-off between the two optimization objectives of energy consumption and task makespan, by adjusting the proportion of the reward of different optimization objectives. This framework could effectively be used for a trade-off of the two optimization objectives of energy consumption and task makespan. Yao et al. [35] proposed an energy-aware scheduling algorithm called EnMORL. Their objective was to simultaneously minimize the makespan and energy consumption while meeting budget constraints. This method is based on RCB and can be applied to other workflow scheduling problems with budget constraints. Singh et al. [36] proposed an efficient metaheuristic approach called energy-efficient workflow scheduling (EOWS) for minimizing makespan and maximizing energy conservation while scheduling workflows. The results of EOWS were better than both HCRO and MPQGA, in terms of makespan, conserved energy, and fitness value. Belgacem et al. [37] proposed a dynamic resource allocation model to meet customer demand for resources with improved responsiveness. Their model also proposes a multiobjective search algorithm called the spacing multiobjective ant lion algorithm (S-MOAL) to minimize both the makespan and the cost of using virtual machines. Its impact on energy consumption with the dynamic management of resources was studied through the use of unused virtual machines (VMs), which indicated that their approach would save energy. Yadav et al. [38] proposed adaptive heuristic algorithms, namely, least medial square regression, for minimizing energy consumption with minimal service level agreement (SLA). The main idea is overloaded host detection and minimum utilization prediction for VM selection from overloaded hosts. Baker et al. [39] implemented a new multcloud service broker called Cloud-SEnergy for locating, invoking, and integrating the most energy-aware services for multcloud service providers. Table 1 summarizes the previous studies on scheduling policies for allocating tasks on processors with different objectives.

3. Problem Statement

Modern computing centers and data centers consume large amounts of energy, which is mainly derived from conventional energy generated from fossil fuels. For example, a medium-sized data center, such as a university data center, consumes about 80,000 kilowatts of electricity.

In this paper, the best heuristic scheduling (BHS) algorithm for allocating resources to the tasks in the queue created by sorting tasks based on heuristic and grasshopper algorithms and selection of the most appropriate method for performing each task based on the importance factor determined by the end-user or service provider is recommended to achieve a suitable solution in a timely and economic manner. The proposed algorithm has a dual-purpose objective function: to reduce energy and makespan consumption according to the importance factor determined by the resource provider or end-user. The symbols used in this paper are summarized in Table 2.

3.1. Problem Models. The S-sequence is first considered as a DAG by evaluating the dependencies between parts of the tasks in real-time. $G = (T, D)$ represents this graph, where $T$ is the set of tasks and $D$ is the set of link analysis. For the task $i, j$, $T$ link analysis in the form $(i \rightarrow j)$ determines the dependence between the tasks $i$ and $j$, so that task $j$ must be executed after the end of task $i$. In this case, $i$ is the parent and $j$ is the child; a node can have several parent nodes. The node $j$ starts to run when all of its parents are fully executed. A cloud platform is considered as a set of nodes $\{n_1, n_2, \ldots, n_n\}$, whereby each node is responsible for managing a set of virtual machines $VM = \{vm_1, vm_2, \ldots, vm_m\}$, where the virtual machine $k$ is given from node $m$ by $vm_k$. Each virtual machine $k$ has a set of processing cores $C_{vm_k} = \{c_{vm_k1}, c_{vm_k2}, \ldots, c_{vm_kp}\}$, in which the core $i$ from the virtual machine $k$ is represented by $c_{vm_ki}$. Finally,
### Table 1: Previous studies on energy-aware task scheduling techniques.

| Authors                        | Proposed method | Target                                                                 |
|-------------------------------|-----------------|------------------------------------------------------------------------|
| Barzegar et al. [2]           | EATSDCD         | Reducing the makespan and energy consumed by the processors to perform the tasks in the DAG |
| Juarez et al. [22]            | MHRA            | Minimize the energy consumption and makespan                           |
| Sofia and GaneshKumar [27]    | NSGA-II         | Minimizing the energy consumption and makespan from the cloud service  |
| Yang et al. [31]              | EISTS           | Balance between optimization of makespan as well as energy consumption |
| Ben Alla et al. [33]          | EATSD           | Reduce the energy consumption of the cloud resources and optimize the makespan under the deadlines constraints |
| Peng et al. [34]              | DQN             | The framework for trade-off optimization energy consumption and task makespan |
| Yao et al. [35]               | EnMORL          | Simultaneously minimize the makespan and energy consumption while meeting the budget constraint |
| Singh et al. [36]             | EEWS            | Minimizing makespan and maximizing energy conservation while scheduling workflow |
| Belgacem et al. [37]          | S-MOAL          | Minimize both the makespan and the cost of using virtual machines     |

### Table 2: Symbols used in this paper.

| Symbol          | Description                                                                 |
|-----------------|-----------------------------------------------------------------------------|
| $c_{mki}$       | Core $i$ from the virtual machine $k$                                        |
| $S_{mki}$       | Set of schedules for all cloud resources                                    |
| $C_{\text{max}}$ | Makespan                                                                   |
| $E_{\text{flow}}$ | Energy flow                                                                 |
| $\alpha$        | Importance factor determined by the end-user or service provider            |
| $\text{task}_{mki}$ | Task $j$ assigned to core $i$ of the virtual machine $k$ in the node $m$ |
| $\text{time}_{\text{start}}$ | Initial time of the task $j$                                                |
| $\text{time}_{\text{setup}}$ | Setup time of the task $j$                                                  |
| $\text{time}_{\text{end}}$ | End time of the task $j$                                                    |
| $T_{\text{Prej}}$ | A set of prerequisite tasks for the task $j$                               |
| $\text{time}_{\text{pre}}$ | Start time of the task $j$ prerequisite                                      |
| $\text{time}_{\text{end pre}}$ | End time of task $j$ prerequisites                                           |
| $\text{timetransfer}$ | Data transfer times required to perform a task that is outside the virtual machine |
| $\text{time}_{\text{exec}}$ | Processing time of a task $j$ in the core $i$ of the virtual machine $k$ in the node $m$ |
| $e_{\text{task}_{mki}}$ | Energy consumed by a task $j$                                               |
| $e_{\text{transfer} \, \text{mk}}$ | Energy consumed by the data transfer of the task $j$                         |
| $e_{\text{vmk}}$ | Energy consumed by the virtual machine $k$                                  |
| $e_{\text{task} \, \text{mk}}$ | Energy consumption by various tasks $j$ in a virtual machine $k$             |
| $e_{\text{nn}}$ | Energy consumption per node                                                 |
| $e_{\text{net}}$ | Average data transfer power in cloud infrastructure                        |
| $\bar{P}_{\text{vmk}}$ | Power proportional to the use of a specific core in a virtual machine       |
| $P_{\text{down} \, \text{vmk}}$ | Power consumed to up a virtual machine                                      |
| $P_{\text{PUD}}$ | Power consumed to turn off the virtual machine                              |
| $P_{\text{UD}}$ | Power consumed by support systems                                           |
| $\text{Run}_{\text{vmk}}$ | Virtual machine setup time                                                  |
| $\text{down}_{\text{vmk}}$ | Virtual machine shutdown time                                               |
| $\text{setup}_{\text{vmk}}$ | Node running time                                                           |
| $\text{set}_{\text{vmk}}$ | Location of grasshopper $i$                                                 |
| $X_i$            | Social interaction between grasshoppers                                       |
| $S_i$            | Gravity force on the grasshopper $i$                                        |
| $A_i$            | Wind movement                                                               |
| $d_{ij}$         | Distance between the grasshopper $i$ and grasshopper $j$                    |
| $d_{ij}^*$       | Unique vector from the grasshopper $i$ to the grasshopper $j$               |
| $f$              | Gravity intensity                                                           |
| $l$              | Size of the absorption length                                               |
| $g$              | Gravitational constant                                                      |
| $u$              | Thrust constant                                                             |
| $N$              | Number of grasshoppers                                                      |
| $u_{\text{ubd}}$ | Upper bound on the dimension $D$                                            |
| $l_{\text{lb}}$  | Lower bound on the dimension $D$                                            |
the scheduling problem involves searching for the S scheduling. This indicates the order in which each task j is performed in the available set of resources. Thus, the scheduling for C_{mki} is shown by S_{mki}, which is presented by equation (1). This shows that the n task is executed to core i of the virtual machine k from the node (j_n \rightarrow C_{mki}). The complete solution S can be represented by equation (2), as a set of schedules for all cloud resources:

\[ S_{mki} = \{ j_1 \rightarrow C_{mki}, j_2 \rightarrow C_{mki}, \ldots, j_n \rightarrow C_{mki} \} \]  
\[ S = \{ S_{mki} \} \]  

In the case of scheduling, the search process for scheduling tasks is guided by two objectives:

1. To improve the energy efficiency by looking for the best positions of tasks in resources such that the energy consumed by the implementation of the whole set of tasks is minimized.
2. To improve executive performance by searching for the allocation of resources that minimize the makespan.

However, a balanced relationship between energy efficiency and makespan provides a solution that minimizes energy consumption and increases makespan. The objective of the proposed scheduling strategy is to reconcile the two objectives. Therefore, we propose a function that combines energy consumption and makespan, with C_max or makespan and E_flow as the energy flow with a coefficient of significance defined by α, to determine for the end-user or resource provider whether energy or makespan is more important. Accordingly, the proposed scheduling problem is modeled as an optimization problem, which seeks a solution that minimizes the objective function shown in the following equation:

\[ \text{Min} \left( \alpha \left( \frac{C_{\text{max}}}{c_{id}} \right) + (1 - \alpha) \left( \frac{E_{\text{flow}}}{E_{id}} \right) \right) \] 

The objective function is composed of two main terms E_{flow} (S) and C_{max} (S), where E_{flow} (S) is the energy consumed by the implementation of solution S and C_{max} (S) is the makespan, defined as the maximum time taken for the last task to leave the system.

To balance the objective function, the adaptive weighted sum method is applied, which introduces a weight factor α, where 0 ≤ α ≤ 1, to indicate which term is more important for the end-users or service providers. However, makespan and total energy flow have different units. The makespan is calculated in seconds, while energy consumption is calculated in watt-hours (Wh). Two factors, c_{sf} and E_{sf}, have been used to normalize energy flow and makespan that are never equal to zero. As explained, the makespan is defined by the last task that leaves the system, and estimating E_{flow} is challenging. The next section provides further details about the model used to estimate E_{flow} for the S schedule.

3.1.1. Energy Model. The objective function presented in the previous section requires the calculation of the exponential and energy for the solution of a given schedule. Through this scheduling solution, an energy-aware scheduler can extract the following times to estimate the makespan and energy consumption:

(i) Start and end time of tasks performed in each specified core.
(ii) Start and end time of the data transfer required for each task.
(iii) Start and end time of virtual machine setup and shutdown.
(iv) Start and end time for each virtual machine.
(v) Start and end time for each node.

Another critical piece of information for energy calculation is the power profile for different sources. For each source, we must calculate the following values:

1. P_{cmki} is the average power consumed by core i of a virtual machine k in node m.
2. P_{up V m k} and P_{down V m k} are the average power consumed by the virtual machine when the service provider creates and destroys the virtual machine k in node m.
3. P_{idle} is the average power consumed by node m in the idle mode. In other words, the power consumed is such that the service provider is not in control of the virtual machines and only the operating system services are running.
4. P_{net nm} is the average power required to transmit data through cloud infrastructure.
5. PUE is the average power consumed by the support system (UPS, coolers, etc.).

Finally, the energy consumed by each parameter can be estimated by multiplying the duration of the different events by the corresponding average amount of power.

3.1.2. Energy Consumed by Each Task. To calculate the energy consumed by task j assigned to the core i of the virtual machine k in the node (task_{mki}), the initial time (time_{init j}), setup time (time_{setup j}), and end time (time_{end j}) of each task must be estimated.

Task Start Time. Each task j has a set of pretasks (T_{pre}) L that must be completed before task j begins. Therefore, the minimum start time for task j depends on the longest end time of one of the previous tasks L (time_{pre j}). Calculated according to the following equation, a task that has no precedent timepre j is equal to zero:

\[ \text{Time}_{pre j} = \text{Max} \left( \text{time}_i \text{L}_{pre} \right) \] 

The task j is the first task assigned to a core, and the start time is directly determined by the preceding end time. However, if the tasks are assigned before this to a data center,
the start time depends on the end time before the task, shown by timeendj.

In this case, the start time of the task is expressed by the maximum value of the most significant preceding end time (timeprej) and the maximum end time of the previously assigned tasks in the same core c_mki, that is, t \in T_{mki}. Finally, j is the first task assigned to the virtual machine k of the node m (Vm_k); the start time depends on the setup time in Vm_k.

If this time is more than the preceding maximum end time, the task start time (timeinitj) will be considered equal to setupVm_k. Calculation of the start time of task j is shown as follows:

\[
\text{time}_{\text{init}}j = \max(\text{time}_{\text{pre}}j, \max(\text{time}_{\text{transfer}}Vm_k), \text{setup}_{Vm_k}),
\]

(5)

**Task Setup Time.** Before starting a task, the data required for the task must be transferred to the source where it is executed. This time is known as the setup time, denoted by time\_setupj. It depends on the proposed scheduling solution because the extent of data transfer depends on the source to which the previous tasks were assigned. Therefore, the setup time for task j is equal to the sum of all the required data transfer times (time\_transferVm_k(f_j)) outside the virtual machine:

\[
\text{time}_{\text{setup}}j = \sum_{l \in T_{pre}Vm_k} \text{time}_{\text{transfer}}Vm_k(f_j).
\]

(6)

**Task End Time.** Task end time (time\_endj) equals total task initial time (time\_initj), task setup time, and task execution time j in core i of the virtual machine k from node m (time\_execj) and is calculated as follows:

\[
\text{time}_{\text{end}}j = \text{time}_{\text{init}}j + \text{time}_{\text{setup}}j + \text{time}_{\text{exec}}j.
\]

(7)

**Energy Consumption of Each Task.** The energy consumed by task j is represented by e_taskmk_k_j, which is estimated by the sum of the energy consumed by the data transfer of a task and the energy consumed by the task processing. The energy consumed by the data transfer is estimated by multiplying the time spent transferring the required data by the average power of the data transfer in the cloud infrastructure (P_net). In contrast, the energy consumed by task processing is calculated by multiplying the processing time of work j with the average power proportional to using a special core in a virtual machine (P_Cmki), using the following equations:

\[
e_{\text{transfer}}mk_j = \text{time}_{\text{setup}}j \times P_{\text{net}Vm_k},
\]

\[
e_{\text{task}}mk_k_j = \text{time}_{\text{exec}}j \times P_{Cmki}.
\]

(8)

(9)

**Energy Consumed by Each Virtual Machine.** The second parameter that must be calculated to estimate the energy flow is the amount of energy consumed by each virtual machine used in the cloud (Vm_k). The energy consumed by the virtual machine k is denoted by e_{Vm_k}, which is calculated as follows:

\[
e_{Vm_k} = \text{setup}_{Vm_k} \times P_{upVm_k} + e_{\text{task}}mk_k + \text{down}_{Vm_k} \times P_{downVm_k}.
\]

(10)

This equation includes the amount of power consumed to start and shutdown the virtual machine k from node m, which is estimated by multiplying the time elapsed for it. Setup and shutdown times of the virtual machine are (setupVm_k, downVm_k), with average power (P_{upVm_k}, P_{downVm_k}). Energy consumption is calculated by the various tasks j in the virtual machine k as follows:

\[
e_{\text{task}}mk_kj = \sum_{l \in T_{mki}} e_{\text{task}}mk_kj.
\]

(11)

**Energy Consumed by Each Node.** The energy consumed by a node (e_{node}) is calculated based on the total amount of energy consumed by virtual machines when they are running and idle. The energy consumed by all of the virtual machines in a node is calculated as

\[
e_{Vm} = \sum_{Vm_k \in Vm} e_{Vm_k}.
\]

(12)

When we calculate the energy flow, the node is shutdown, all nodes running at the same time are equal to the makespan. Accordingly, the makespan of the node is shown as

\[
\text{Run}_{vm} = C_{\text{max}}.
\]

(13)

Once the makespan of the node is estimated, we can estimate the amount of energy consumption per node as follows:

\[
e_{mn} = \text{Run}_{vm} \times P_{\text{idle}} + e_{vm}.
\]

(14)

**3.1.3. Makespan and Energy Flow.** When we calculate the energy required for tasks, virtual machines, and nodes, we must calculate C_{max} and E_{flow}. Makespan can be calculated using equation (15). It is based on the sum of the end times of the last task j performed in the cloud and the shutdown time of the virtual machine in which this task is performed:

\[
C_{\text{max}} = \max(\text{time}_{\text{end}}j + \text{down}_{Vm_k}).
\]

(15)

The energy flow (E_{flow}) is calculated as equal to the sum of the energy consumed by each node in power consumed by the support systems (e.g., the cooling system and UPS), which is calculated using the following equation:

\[
E_{\text{flow}} = PUD \times \left(\sum_{mn \in N} e_{mn}\right).
\]

(16)

**4. Proposed Method.**

The best heuristic scheduling (BHS) is to schedule resources and allocate them to tasks and thus generate the solution of
the high-quality scheduler in a real period. The BHS algorithm offers a step-by-step solution to achieve the desired goal. The stages of the proposed heuristic algorithm are shown in Figure 1.

The ideal heuristic scheduling algorithm receives the DAG, which includes the set of tasks to be performed on cloud processors as input. The algorithm automatically analyzes the DAG and implements a set of free tasks that are not required to be performed.

This algorithm automatically analyzes the DAG and a set of free tasks that do not have prerequisites or with prerequisites that have been executed and can be executed in parallel with specifying the implementation. The main algorithm (BHS) then proceeds as follows: first, the tasks are sorted based on the heuristic methods and the grasshopper algorithm. They are placed in the corresponding list of that method. This step generates five list types, before progressing to the next step to allocate resources to the task. In this step, the resources assign necessary tasks by resource allocation cost function algorithm (RACF), according to energy profiles of the tasks based on the amount of memory and the core to perform each task. The resource is allocated, and the amount of objective function for each of the five lists formed in the previous step is calculated based on the importance factor determined by the end-user or service provider. The next step is to estimate the amount of makespan and energy consumed for each of the five lists. In the last step, using a roulette wheel, the amount of the calculated objective function for each of the five lists formed is considered to be calculated based on the importance factor determined by the end-user or service provider. One of them has the amount of the objective function to perform the desired task. More optimal (minimum) value is selected.

### 4.1. Grasshopper Algorithm

The grasshopper belongs to the insect family. Grasshoppers are known as pests because they damage agricultural products. Although they are usually found in nature individually, they can also be found together in one of the largest groups of all creatures. The size of these groups may be enormous and can be a nightmare for farmers. The mathematical model of the grasshopper algorithm mimics the behavior of grasshoppers in nature to solve the optimization problem. The simulation results show that the locust algorithm is able to provide better results compared to known algorithms and recent literature. Simulation results on real problems also proved that the grasshopper algorithm is able to solve real problems with unknown space [40].

#### 4.1.1. Explanation of Grasshopper Behavior by Mathematical Relations

The mathematical model used to simulate the behavior of grasshoppers is presented in the following equation:

$$X_i = S_i + G_i + A_i$$  \hspace{1cm} (17)

$X_i$ is the location of the grasshoppers $i$, $S_i$ is the social interaction between grasshoppers, $G_i$ is the gravitational force between grasshoppers, and $A_i$ is the movement of the wind. Since grasshopper nymphs only walk on the ground and do not jump, the gravity of the Earth does not affect them. While wind movement has the most significant effect on nymphs, we can ignore them and consider only adult grasshoppers:

$$S_i = \sum_{j=1}^{N} S(d_{ij}) \vec{d}_{ij},$$

where $d_{ij}$ is the distance between grasshopper $i$ and grasshopper $j$ and is calculated as $d_{ij} = |x_j - x_i|$, $S$ is a function that shows the power of social interaction in a high relation, and $\vec{d}_{ij} = ((x_j - x_i)/d_{ij})$ is a unique vector from grasshopper $i$ to grasshopper $j$. The $S$ function, which represents social interaction, is calculated by

$$S(r) = f e^{\frac{(r-l)^2}{2l^2}} - e^{-r},$$

where $f$ is the gravity intensity and $l$ is the length of the absorption. The values of $f$ and $l$ should be 1.5 and 0.5, respectively, to obtain an optimal answer.

The gravitational relation between the grasshoppers is also calculated as

$$G_i = -g \vec{e}_{g}.$$  \hspace{1cm} (20)

The gravitational constant is $g$, which is usually considered to be 9.8 or 10, and $\vec{e}$ indicates the unit vector toward the center of the Earth. The wind displacement relation is also calculated from

![Figure 1: The proposed BHS method.](image-url)
8 Scientific Programming

where \( u \) is the thrust constant and \( \vec{w}_u \) is the unit vector that determines the direction of the wind.

The sum of the relations mentioned above determines the value of \( X \) or the next movement of the grasshopper:

\[
X_i = \sum_{j=1}^{N} S \left( \left| x_j - x_i \right| \right) \frac{x_j - x_i}{d_{ij}} - g \vec{v}_g + u \vec{w}_u,
\]

\[\text{Equation (22)}\]

\[e^{-r} - s(r) = fe^{(-r/l)} \] and \( N \) is the number of grasshoppers.

The modified version of this relationship is intended to optimize issues:

\[
X_i = C \left( \sum_{j=1}^{N} \frac{ubd_d - lbd_d}{2} S \left( \left| x_j - x_i \right| \right) \frac{x_j - x_i}{d_{ij}} \right) + \bar{T}_d,
\]

\[\text{Equation (23)}\]

where \( ubd \) is the upper bound in dimension \( D \), \( lbd \) is the lower bound in dimension \( D \), \( s(r) = fe^{(-r/l)} - e^{-r} \), \( T_d \) is the amount of \( D \) after the target (the best solution that has been determined up to that point), \( C \) is a decreasing coefficient, and \( S \) is similar to equation (17).

To balance the search and increase the optimization, it is necessary to reduce the \( C \) parameter proportionally or the number of iterations. In this mechanism, as the number of repetitions increase, optimization increases. The \( C \) coefficient reduces the comfort zone by the number of repetitions and is calculated as

\[
C = C_{\text{max}} - I \frac{C_{\text{max}} - C_{\text{min}}}{L},
\]

\[\text{Equation (24)}\]

where \( C_{\text{max}} \) has the highest value, \( C_{\text{min}} \) has the lowest value, \( I \) represents the current repetition, and \( L \) represents the maximum number of repetitions.

### 5. Evaluation of the Proposed Method

Data analysis is particularly important for the validity of hypotheses for any research. The simulation environment used to evaluate the proposed method is the MATLAB tool. We have installed the MATLAB tool on an Asus notebook with an Intel core i7- A540UP 2.4 GHz CPU with 8 cores and 4 GB memory. In this section, we review the results of implementing the proposed method.

#### 5.1. Evaluation Parameters

To evaluate the proposed method, four virtual machines from Intel and AMD were used, with characteristics as listed in Table 3. In addition, tasks with parameters having the values listed in Table 4 are used.

#### 5.2. Best Heuristic Scheduling (BHS) Algorithm

The BHS algorithm is designed to identify the optimal scheduling of tasks assigned to the cloud and to estimate the optimal possible amount of energy for consumption and the makespan based on the importance factor determined by the end-user or service provider. The pseudocode of BHS is shown in Algorithm 1. This algorithm includes the number of search factors or number of grasshoppers, the number of repetitions of the algorithm, the lower bound and the upper bound, the number of dimensions and features that the grasshopper is supposed to optimize. The resource allocation algorithm and the estimation of the objective function are expressed in Algorithm 2. The virtual machines are received as input and in return for the optimal value of the objective function, and the type of sorting returns the tasks to be executed as output. In line 2, the grasshoppers are given an initial value in such a way that \( N \), or the same number of grasshoppers, is considered to be a position equal to the set of tasks to which the resource is to be assigned. Lines 3 to 5 are investigated based on the objective function to determine whether the implementation of line 2 is appropriate. Lines 6 to 19, based on the grasshopper algorithm formulas, calculate the distance between the grasshoppers considered in the previous section until reaching the target grasshopper that has the lowest possible value, and select the best grasshopper, which represents the best way to perform tasks in this way. From lines 20 to 24, resources are allocated to the lists formed by the grasshopper method and four heuristic methods using the RACF algorithm, and the value of the objective function is calculated for each list. In line 25, out of five values of the objective function obtained in the previous section, the most optimal value should be selected, using a roulette cycle for each of the five possible lists to be considered. The best method (the method that has the lowest amount for the objective function) is then selected to perform each task. In lines 26 to 32, one of the five lists from the previous section is selected since a list of tasks is provided while the number of tasks is required. The number of elements in the identifiers list is then placed in an array called the ID. Line 34 adds one to the number of repetitions of the algorithm, and this cycle continues until the best possible response is found.
5.3. Resource Allocation Cost Function. Resource allocation cost function (RACF) algorithm is used to allocate resources to tasks and calculate the amount of objective function expressed by Algorithm 2. In this algorithm, the list of tasks, resources, and significance coefficient is received as input and the amount of objective function, energy consumption, and makespan is estimated and finally returned as output. The first line of this algorithm is related to the introduction of the function, which receives a list of tasks, virtual machines, and importance factors and returns the amount of the objective function, energy consumed, and makespan as the output of the algorithm. In line 2, there are tasks that have not yet been processed or tasks that have been reserved and that are running, as long as these two modes are in place, the steps are repeated. Line 3 investigates whether resources should be allocated to tasks or should wait for a task to be completed. In line 4, if it has to be allocated to the resource task, it will consider that task as the current task. Lines 5 and 6 examine the first to the last virtual machine, to find the appropriate virtual machine with the required amount of RAM and core to perform the task, and most importantly, that virtual machine is switched on. In lines 7 and 8, the virtual machine assigns the desired task. Line 9 calculates the makespan (end time) of the task. In line 10, the task enters the reservation list. In line 11, the makespan is calculated. In lines 12 to 16, the energy consumed during data transfer, the energy consumption of each task, the energy consumption of each virtual machine, the energy consumption of each node, and finally energy consumption are calculated according to the equations described in Section 4. Lines 21 to 27 proceed when the task at hand is completed, and the resources allocated to it must be released. At the end of line 31, the amount of the objective function expressed in equation (3) is calculated.

5.4. Evaluation Results. On evaluating the proposed method, heuristic methods and grasshopper algorithms were used to sort the tasks each time they were performed. Based on the proposed BHS algorithm, a sorting method was selected for each task, resulting in the most optimal response. This has improved the proposed method compared to the MHRA method.

5.4.1. Effect of the Importance Factor on Energy Consumption and Makespan. In this section, for 1000 tasks, we changed the importance factor (α) from 0.0 to 1.0 in each step by 0.1. According to the results of the LPT exploratory method shown in Figure 2, performing this task is more appropriate than other methods and provides more optimal results.

As shown in Figure 2, when the importance factor reaches 1.0, energy consumption increases to the maximum possible value of 3584 Wh, while the makespan decreases to the minimum possible value of 15,000 sec. When the importance factor reaches 0.0, the energy consumption decreases to the minimum possible value of 2541 Wh, while the makespan increases to the maximum possible value of 16,298 sec. Therefore, as the importance factor increases from 0.0 to 1.0, the amount of energy consumption increases, while the makespan decreases. By increasing the importance factor, more resources are used to speed up the tasks. It can be concluded that when the energy consumption is more important for the end-user or service provider, the importance factor should be set to 0.0. Otherwise, when makespan is more important, the end-user or service provider must set the importance factor to 1.0. As a result, proper selection of the importance factor is crucial.

5.4.2. Choice of Importance Factor When Both Items (Energy Consumption and Makespan) Are Important to the End-User or Service Provider. In this section, the amount of energy consumption and makespan with different importance factors are reviewed and compared to determine the importance factor according to Figure 3, which shows the makespan and energy consumption in an acceptable state for the end-user or service provider.

As mentioned in the previous section, when the energy consumption is more important for the end-user or service provider, the importance factor is equal to 0.0, and when makespan is more important, the importance factor is equal to 1.0. According to Figure 3, with an importance factor of 0.7, the amount of energy consumed is equal to 3017.9 Wh, and the makespan is 15,399 sec. The graph of both cases somewhat comes down, which indicates that the amount of energy consumed and makespan are both reduced. In summary, when both power consumption and makespan are important to the end-user or service provider, the importance factor should be 0.7.

5.4.3. Comparison of the Proposed BHS Method with MHRA. In this section, we review and compare the energy consumption and the proposed BHS method with the MHRA method. For this purpose, the implementation of 1000 tasks with different coefficients of importance and the LPT heuristic method for two scheduling methods was analyzed.

| Parameter       | INTEL   | AMD     |
|-----------------|---------|---------|
| Setup time      | 175 s   | 85 s    |
| Shutdown time   | 3 s     | 2 s     |
| Number of cores | 4       | 4       |
| ROM             | 4000 MB | 4000 mb |
| Hard            | 200 MB  | 200 mb  |
| Power core      | 9.74 W  | 11.2 W  |
| Setup power     | 9.49 W  | 18.24 W |
| Shutdown power  | 9.49 W  | 18.24 W |
| Unemployment power | 175.76 W | 115.30 W |
| Transfer power  | 30.04 W | 27.56 W |
| Support power   | 1.2 W   | 1.2 W   |

| Parameter | Value            |
|-----------|------------------|
| Processing time | 80–100 s    |
| Number of cores | 1–4      |
| ROM       | 100–1000 MB     |
| Hard drive rate | 100–1000 MB |
The results are shown in Figures 4 and 5 and suggest that the proposed BHS method returns more optimal results than the MHRA method.

According to the results obtained in the previous sections, the lowest energy consumption with an importance factor of 0.0 for the BHS method is equal to 2500 Wh and for the MHRA method is equal to 3113.6 Wh. It can be concluded that the BHS method, using the least amount of available resources, causes significant energy storage and is suitable for reducing energy consumption.

When we consider the importance factor as 1.0, the makespan value has the lowest value, which is equal to 15,000 seconds in the BHS method and 11,425.6 seconds in the MHRA method, which has increased by 31.28%. When the importance factor is 0.0, makespan has the highest value, which is 16,298 seconds in the BHS method and 37,138 seconds in the MHRA method, which is reduced by 56.12%. With a change in the importance factor, according to the graph, the makespan changed less with the BHS method than with the MHRA method. This indicates that, in the MHRA method, with a decreasing importance factor that approaches 0.0, the makespan increases significantly. When the importance factor is equal to 0.0, it increases by about 73.1% compared to its value with an importance factor of 1.0. However, this is not the case in the BHS method, and the rate of change from the importance factor of 0.0 to 1.0 is about 7.9%. As a result, the BHS method demonstrates a significant
Input: list of task, resources, and alpha
Output: assigning the best resources to tasks and estimating the amount of objective function according to alpha

1. \(f_{\text{RACF}}(\text{list,VM,} \alpha)\)
2. While \(I \leq \text{length(list)} \land \text{isempty(reserved)}\)
3. If \(i \leq \text{length(list)}\)
   4. \(\text{CurrentTask} = \text{list}(i)\)
   5. For \(j = 1: \text{length(VM)}\)
   6. If \(\text{CurrentTask.Ram} \leq \text{VM}(j).\text{Ram} \land \text{CurrentTask.Core} \leq \text{VM}(j).\text{Core} \land \text{VM}(j).\text{On} = \text{true}\)
      7. \(\text{VM}(j).\text{Ram} = \text{VM}(j).\text{Ram} - \text{CurrentTask.Ram}\)
      8. \(\text{VM}(j).\text{Core} = \text{VM}(j).\text{Core} - \text{CurrentTask.Core}\)
      9. \(\text{time endCurrentTask} = \text{time initCurrentTask} + \text{time setupCurrentTask} + \text{time execCurrentTask}\)
   10. \(\text{reserved} = [\text{reserved}; [i j \text{time endCurrentTask} \text{CurrentTask}]]\)
   11. \(C_{\text{max}} = \text{Max} (\text{time endCurrentTask} + \text{down j})\)
   12. \(e_{\text{transfer}} = \text{transfer time setupCurrentTask} \times P_{\text{transfer}}\)
   13. \(e_{\text{CurrentTask}} = \text{transfer} \times (\text{time exec j} \text{VM}(j).\text{Core} \times P_{\text{Core}})\)
   14. \(e_{\text{run}} = \text{setup j} \times P_{\text{runj}} + e_{\text{CurrentTask}} + \text{down j} \times P_{\text{down j}}\)
   15. \(E_{\text{flow}} = P_{\text{UD}} \times (\sum_{n} E_{\text{run}})\)
   16. \(i = i + 1\)
   17. End if;
18. End for;
19. End if;
20. End if;
21. For \(r = 1: \text{size(reserved, 1)}\)
22. If \(\text{reserved}(r, 3) < \text{time}\)
23. \(\text{VM}(j).\text{Ram} = \text{VM}(j).\text{Ram} + \text{list(reserved}(r, 1)).\text{Ram}\)
24. \(\text{VM}(j).\text{Core} = \text{VM}(j).\text{Core} + \text{list(reserved}(r, 1)).\text{Core}\)
25. \(\text{ID} = [\text{ID r}]\)
26. End if;
27. End for;
28. \(\text{Reserved} (\text{ID},:) = []\)
29. \(\text{time} = \text{time} + 1\)
30. End while;
31. \(f = \alpha(C_{\text{max}}/C_{\text{s}}) + (1 - \alpha) (E_{\text{flow}}/E_{\text{f}})\)
32. End

**Algorithm 2**: Resource allocation cost function (RACF).

(a) Makespan (seconds) vs. Importance factor \(\alpha\)

(b) Energy consumption (wh) vs. Importance factor \(\alpha\)

Figure 2: Impact of the importance factor on energy consumption and makespan in the LPT method.
reduction in energy consumption and makespan compared to the MHRA method when the importance factor is 0.0.

6. Time Complexity Analysis

We want to analyze the time complexity of algorithms presented in the previous sections.

6.1. Analysis of BHS Algorithm. This algorithm is designed to find the best task schedule based on the importance factor determined by the user or service provider. Lines 3 to 5 run the number of grasshoppers, i.e., |N| times of the RACF function to allocate resources to tasks and calculate the target function. Also, the innermost for loop with index j (lines 12 to 16) is |N| times performing the next grasshopper position estimation operation according to its current position. For loop with index k (lines 11 to 18) n/2 times, perform checking and sorting operations. Lines 20 to 24 also call the RACF function to calculate the value of the target function of the generated lists, whose complexity is |n| × (|m| + |m|). Given that these instructions are in the for loop with index i, where |N| repeats this operation, so the number of executions is equal to |N| × (|N| + |N| + |n| × (|m| + |m|)). Also, the while loop in line 7 has a max iteration number of |L| times performing the entire operation above, so the number of runs is equal to |L| × (|N| × (|N| + |N| + |n| × (|m| + |m|))), and therefore, the complexity of the BHS algorithm is equal to \( O(L|N^2| + |N| N_m)) \).

6.2. Analysis of RACF Algorithm. This algorithm is used to allocate resources to the tasks in the lists created in the BHS algorithm and to calculate the value of the objective function to execute each list. Assuming that the number of virtual machines is equal to \( m \), then for loop with index j (lines 5 to 19), which performs the operation of allocating resources to the tasks in the list and calculating the amount of energy consumption and makespan, it is executed \(|m| \) times, and for loop with index r, which performs the operation of freeing up the resources allocated to the tasks that are on the reserve list and have been completed, it is also executed \(|m| \) times. Given that both for loops are in an external while loop and assuming that the number of list elements is \( n \), the while loop is \(|n| \) runs when the total number of operations is equal to \(|n| \times c(|m| + |m|) \). As a result, the complexity of the RACF algorithm is equal to \( O(|mnl|) \).

6.3. Analysis of MHRA Algorithm. The MHRA algorithm in the paper [22] consists of 5 nests for loops. The for loop in
line 2 is executed in the number of heuristic methods $K = 4$ times. Also, the number of iterations in two for loops in lines 3 and 5 is, respectively, 10 and $|n|$ times. $|n|$ is equal to the number of nodes in the DAG. In addition, the for loop in line 7 where task sorting operations can be done at times $|n|$ is based on the exploratory method. Also, the for loop in line 9 has the number of cores $|m|$ times performing resource allocation operations to tasks. Therefore, the number of executions is equal to $4 \times 10 \times (|n| \times (|n| \times |m|))$, and therefore, the complexity of the MHRA algorithm is equal to $O(|n|^2 |m|)$.

7. Conclusions and Future Work

In this paper, we have presented a dual-objective scheduling algorithm that is aware of makespan and energy consumption in order to allocate resources to tasks and to sort tasks, based on heuristic methods and grasshopper algorithm. Furthermore, the most appropriate method to execute each task is selected based on the importance factor determined by the end-user or service provider, set by using the roulette wheel in the cloud. The proposed BHS algorithm minimizes the target function according to this importance factor, taking into account the start times, setup times, and end times for virtual machines and energy profiles.

To implement the desired algorithm, MATLAB simulator was used with a DAG generated for 1,000 tasks having different coefficients of importance and four virtual machines from Intel and AMD. The results of the assessment indicate that, to execute any task based on the importance factor determined by end-user or service provider, a specific type of heuristic or grasshopper method is appropriate. The response is more efficient compared to the rest of the methods. It can also be concluded that when energy consumption is more important for the end-user or service provider, they must set the importance factor equal to 1.0. When the makespan is more critical, the end-user or service provider must set the importance factor equal to 0.0, and when both the energy consumption and the makespan are critical, they must set the importance factor equal to 0.7. Therefore, determining the importance factor is very important. The evaluation results show that the proposed algorithm has reduced energy consumption by 19.71% compared to MHRA and has reduced the makespan by 56.12% when energy consumption is of high importance. Overall, selecting the optimal method for sorting and performing each task can significantly reduce the makespan and amount of energy consumed, as well as the costs involved.

Future research should investigate parameters such as reliability, performance enhancement, and security enhancement for each task, based on the importance factor set by the end-user or service provider.

Data Availability

The data used to support the findings of this study are available on request to the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Youth Program of National Natural Science Foundation of China under grant no. 11701062.

References

[1] R. Mandal, M. K. Mondal, S. Banerjee, and U. Biswas, “An approach toward design and development of an energy-aware VM selection policy with improved SLA violation in the domain of green cloud computing,” The Journal of Supercomputing, vol. 76, no. 9, pp. 7374–7393, 2020.
[2] B. Barzegar, H. Motameni, and A. Movaghar, “EATSDCD: a green energy-aware scheduling algorithm for parallel task-based application using clustering, duplication and DVFS technique in cloud datacenters,” Journal of Intelligent & Fuzzy Systems, vol. 36, no. 6, pp. 5135–5152, 2019.
[3] Z. Zhou, J. Abawajy, M. Chowdhury et al., “Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms,” Future Generation Computer Systems, vol. 86, pp. 836–850, 2018.
[4] L. Wang, S. U. Khan, D. Chen et al., “Energy-aware parallel task scheduling in a cluster,” Future Generation Computer Systems, vol. 29, no. 7, pp. 1661–1670, 2013.
[5] K. Bilal, S. U. Khan, and A. Y. Zomaya, “Green data center networks: challenges and opportunities,” in Proceedings of the 2013 Conference on Frontiers of Information Technology (FIT), pp. 229–234, Islamabad, Pakistan, December 2013.
[6] R. Yadav and W. Zhang, “MeReg: managing energy-SLA tradeoff for green mobile cloud computing,” Wireless Communications and Mobile Computing, vol. 2017, Article ID 6741972, 11 pages, 2017.
[7] R. Yadav, W. Zhang, H. Chen, and T. Guo, “MuMs: energy-aware VM selection scheme for cloud data center,” in Proceedings of the 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), pp. 132–136, IEEE, Lyon, France, August 2017.
[8] V. Ali and M. E. Khodayar, “Energy-aware cloud computing,” The Electricity Journal, vol. 31, no. 2, pp. 40–49, 2018.
[9] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, “The cost of a cloud,” ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 68–73, 2008.
[10] S. Mustafa, B. Nazir, A. Hayat, A. u. R. Khan, and S. A. Madani, “Resource management in cloud computing: taxonomy, prospects, and challenges,” Computers and Electrical Engineering, vol. 47, pp. 186–203, 2015.
[11] P. A. Sanjay and K. Mulhiah, “Green computing strategies for competitive,” Advantage and Business Sustainability, vol. 16, 2018.
[12] D. Kliazovich, B. Pascal, and S. U. Khan, “GreenCloud: a packet-level simulator of energy-aware cloud computing data centers,” The Journal of Supercomputing, vol. 62, pp. 1263–1283, 2012.
[13] S. Yassa, R. Chelouah, K. Hubert, and B. Granado, “Multi-objective Apageroaeh for energy-aware workflow scheduling in cloud computing environments,” The Scientific World Journal, vol. 2013, 13 pages, Article ID 350934, 2013.
[14] J. G. Koomen, Growth in Data Center Electricity use 2005 to 2010, pp. 1–24, Stanford University, Stanford, CA, USA, 2011.
[15] A. Verma and S. Kaushal, "A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1–19, 2017.

[16] F. Zhang, J. Cao, K. Li, S. U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Generation Computer Systems*, vol. 37, pp. 309–320, 2014.

[17] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: understanding the runtime effects of frequency scaling," in *Proceedings of the Conference on Supercomputing*, New York, NY, USA, June 2002.

[18] V. Tiwari, S. Malik, and A. Wolfe, "Compilation techniques for low energy: an overview," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, San Diego, CA, USA, October 1994.

[19] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[20] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp. 141–147, 2014.

[21] W. Chen, R. F. da Silva, E. Deelman, and R. Sakellariou, "Using imbalance metrics to optimize task clustering in scientific workflow executions," *Future Generation Computer Systems*, vol. 46, pp. 69–84, 2015.

[22] F. Juarez, J. Ejarque, and M. B. Rosa, "Dynamic energy-aware scheduling for parallel task-based apagelation in the cloud computing," *Future Generation Computer Systems*, vol. 78, no. 1, pp. 257–271, 2018.

[23] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time constrained workflow tasks in DVFS-enabled cloud environment," *Simulation Modelling Practice and Theory*, vol. 87, pp. 311–326, 2018.

[24] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, and S. Hu, "Minimizing cost andMakespan for workflowschedulingincloud environment," *Simulation Modelling Practice and Simulation*, vol. 87, pp. 311–326, 2018.

[25] R. Yadav, W. Zhang, O. Kaiwartya, P. R. Singh, I. A. Elgendi, and Y. Tian, "Adaptive energy-aware algorithms for minimizing energy consumption and SLA violation in cloud computing." *IEEE Access*, vol. 6, pp. 55923–55936, 2018.

[26] A. S. Sofia and P. GaneshKumar, "Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-ii," *Journal of Network and Systems Management*, vol. 26, no. 2, pp. 463–485, 2018.

[27] G. L. Stavrinides and H. D. Karatza, "An energy-efficient, QoS-aware and cost-effective scheduling apagelation for real-time workflow apagelations in cloud computing systems utilizing DVFS and apageroximate computations," *Future Generation Computer Systems*, vol. 96, pp. 216–226, 2019.

[28] A. Al-Dulaimy, W. Itani, J. Taheri, and M. Shamseddine, "bwSlicer: a bandwidth slicing framework for cloud data centers," *Future Generation Computer Systems*, vol. 112, pp. 767–784, 2020.

[29] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A Deep learning approach for energy efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633, 2019.

[30] Y. Yang, X. Lu, H. Jin, and X. Liao, "A stochastic task scheduling algorithm based on importance-ratio of makespan to energy for heterogenous parallel systems," in *Proceedings of the 2015 IEEE 17th International Conference on High Performance Computing and Communications, 2015 IEEE 7th International Symposium on Cybersecurity Safety and Security, and 2015 IEEE 12th International Conference on Embedded Software and Systems*, IEEE, New York, NY, USA, pp. 390–396, August 2015.

[31] M. Voorneveld, "Characterization of pareto dominance," *Operations Research Letters*, vol. 31, no. 1, pp. 7–11, 2003.

[32] S. Ben Alla, H. Ben Alla, A. Touhafi, and A. Ezzati, "An efficient energy-aware tasks scheduling with deadline-constrained in cloud computing," *Computers*, vol. 8, no. 2, 2019.

[33] V. Tiwari, S. Malik, and A. Wolfe, "Compilation techniques for low energy: an overview," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, San Diego, CA, USA, October 1994.

[34] F. Zhang, J. Cao, K. Li, S. U. Khan, and K. Hwang, "Multi-objective scheduling of many tasks in cloud platforms," *Future Generation Computer Systems*, vol. 37, pp. 309–320, 2014.

[35] A. Miyoshi, C. Lefurgy, E. Van Hensbergen, R. Rajamony, and R. Rajkumar, "Critical power slope: understanding the runtime effects of frequency scaling," in *Proceedings of the Conference on Supercomputing*, New York, NY, USA, June 2002.

[36] V. Tiwari, S. Malik, and A. Wolfe, "Compilation techniques for low energy: an overview," in *Proceedings of 1994 IEEE Symposium on Low Power Electronics*, San Diego, CA, USA, October 1994.

[37] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing," *Future Generation Computer Systems*, vol. 28, no. 5, pp. 755–768, 2012.

[38] C.-M. Wu, R.-S. Chang, and H.-Y. Chan, "A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters," *Future Generation Computer Systems*, vol. 37, pp. 141–147, 2014.

[39] W. Chen, R. F. da Silva, E. Deelman, and R. Sakellariou, "Using imbalance metrics to optimize task clustering in scientific workflow executions," *Future Generation Computer Systems*, vol. 46, pp. 69–84, 2015.

[40] F. Juarez, J. Ejarque, and M. B. Rosa, "Dynamic energy-aware scheduling for parallel task-based apagelation in the cloud computing," *Future Generation Computer Systems*, vol. 78, no. 1, pp. 257–271, 2018.

[41] M. Safari and R. Khorsand, "Energy-aware scheduling algorithm for time constrained workflow tasks in DVFS-enabled cloud environment," *Simulation Modelling Practice and Theory*, vol. 87, pp. 311–326, 2018.