Chapter 6

A hybrid scheduling algorithm with virtual machine consolidation

This chapter[1] *proposes a hybrid scheduling algorithm for workflows in a cloud environment that addresses multi-objectives such as energy reduction, maximizing resource utilization, optimizing numbers of VM migrations, and active cloud servers (physical machines). This algorithm runs in two phases task scheduling and VM consolidation. In the first phase, the task with maximum execution length is mapped to the virtual machine that will perform it with the minimum energy. The second phase contains VM consolidation is a prominent NP-hard problem. The VMC phase categorizes the physical hosts into the normal load, under-loaded and overloaded hosts based on CPU utilization. Double threshold values are used for this purpose. VMs from underloaded and overloaded hosts are migrated to normally loaded hosts. For the VMC phase, we used a nature-inspired meta-heuristic approach called the Water Wave Optimization (WWO) algorithm, which finds a suitable migration plan to reduce the energy consumption by increasing the overall resource utilization and switch off idle hosts after migrating its VMs to a suitable target host. The efficiency of our proposed method evaluated using the WorkflowSim simulation tool with five different real-world scientific workloads.*

[1] Medara, Rambabu, and Ravi Shankar Singh. "Energy-aware workflow task scheduling in clouds with virtual machine consolidation using discrete water wave optimization." *Simulation Modelling Practice and Theory* 110 (2021): 102323.

6.1 Introduction

Over the last few years, the cloud technology [133] has created tremendous potential due to its flexible and pay-as-you-go model. It allows the users to access the cloud services from anywhere, at any time through the Internet. As a growing number of organizations seek to become significant players in today's data-driven economy, cloud data centers remain one of the essential infrastructures. In the cloud data center, virtualization technology has been widely used to virtualize the resources and support the on-demand cloud service paradigm. It provides opportunities for simultaneously running multiple Virtual Machines (VMs) within the infrastructure of a single Physical Machine (PM). There is a massive increase in computing demand in cloud data centers due to the rapid increase in cloud applications. As a result, the cloud data center becomes unsustainable due to the consumption of higher energy in placing input requests on large-scale resources. The cloud services are interrelated with the Service Level Agreements (SLAs). Hence, maintaining a trade-off between energy utilization and performance is essential. Several scientific applications such as chemistry, physics, earth science, astronomy, computer science, bioinformatics, etc., contain a large number of precedence constraint tasks. These scientific applications are extremely complex with different sizes of tasks. The cloud computing environment provides services with different QoS levels to fulfill the workflow application requirements of enormous availability and higher computational energy [2].

Workflow is a set of tasks that are placed in a proper sequence to perform the execution of large-scale real-time applications, which simplifies the processing complexity and management of applications. In workflow DAG, nodes refer to the computational tasks, and edges refer to the precedence constraints between the tasks. To handle the significant increase in the demand for workflow tasks and make sure minimal energy cost, the service providers necessitate the energy-efficient management of the cloud resources. Hence, the existing works have focused on the optimization approaches that rely on the response time and energy without violating the SLAs. In the cloud computing environment, scheduling the tasks to the resources with the satisfaction of QoS requirements is a critical task, especially for complex applications such as workflows [36]. Recently, several researchers have much attention to effectively utilizing cloud resources and reducing energy consumption. Most notably, the workflow scheduling research works [134] [135] have employed the heuristic methods to resolve the constraints in workflow scheduling. Nevertheless, energy-aware

workflow scheduling without resource wastage is still in its infancy stage, requiring few efforts to obtain an optimal solution. Thus, this chapter targets to develop the energy-aware workflow scheduling with VM consolidation without compromising the performance.

The rest of this chapter is structured as follows. The related work is presented in Section 6.2. The proposed system models such as the application model, data center model, and energy model are outlined in Section 6.3. Then Section 6.4 presents the proposed algorithm implementation. Performance evaluation of the algorithm is explained in Section 6.5 and finally, Section 6.6 presents the conclusion and future works.

6.2 Related Work

Currently, there has been a serious consideration in evolving models for scheduling workflow tasks and energy-aware resource management in clouds. Primarily it depends on diverse factors like user request arrival rate, cloud resources availability, and the workload of tasks.

The VM consolidation problem is extensively studied in cloud data centers. Few recent meta-heuristic works on virtual machine consolidation such as genetic algorithm (GA) [136], ant colony optimization (ACO) [19] [137] [138] [139], and particle swarm optimization (PSO) [140] [141] based algorithms efficiently reduced energy consumption. A gravitational effect-based VM placement approach in [142] significantly reduced energy consumption by evaluating computer room attributes to carry out the VM deployment. Predictive-based VM placement techniques in [143] saved significant energy by reducing the active PMs while maintaining system performance. A clustering-based VM placement technique proposed in [144] efficiently reduced the execution time but not considered the energy efficiency objective. An ACO-based VM scheduling proposed in [123] uses vector algebra for consolidating the VMs which reduces the energy utilization and maximizes resource utilization. However, these algorithms not used scientific workflows for performance evaluation.

Many workflow scheduling approaches in the cloud environment considered optimization parameters such as makespan and cost of the application. Zhou, Xiumin, et al. [107] proposed a fuzzy dominance-based Heterogeneous Earliest Finish Time (HEFT) technique

to minimize cost and makespan of workflows in real cloud resources by considering the real pricing models. DAG splitting technique in [108] for large-scale workflow task scheduling reduces the monetary cost by increasing the VM usage. A list-based approach in [92] gives the best makespan. It is observed that most of these works consider bi-objective optimization problems but not addressed the energy objective.

To reduce the energy utilization of the workflow task scheduling in clouds, recent research works have studied different techniques such as resource hibernation, dynamic power management, clustering, etc. The authors in [49] proposed an energy-aware approach for time-constrained workflow in clouds using DVFS (dynamic voltage and frequency scaling) approach. It also considered cost and resource utilization objectives by satisfying SLAs. A Game-Score simulator proposed in [114] for scheduling parallel tasks in the cloud. It trade-off between energy consumption and makespan. Li, Chunlin, et al. [44] proposed workflow scheduling in a distributed cloud which maximizes resource utilization by calculating task running time by considering service status. A PSO-based multi-objective algorithm in [115] minimizes cost, makespan, and maximizes resource utilization while maintaining system reliability. The authors in [77] proposed a PESVMC algorithm that combined the VMC problem with the VM scheduling to schedule the workflow tasks. Li, Zhongjin, et al. [38] proposed a cost and energy-efficient algorithm which uses sequence and parallel task merging, VM reuse, and slack techniques for saving energy.

In the heterogeneous cloud computing infrastructure, scheduling the workflow applications is a challenging task due to the necessity of satisfying the QoS requirements during the task scheduling process. The conventional workflow scheduling research works have focused on optimizing the time or cost, regardless of energy consumption. The lack of satisfying the QoS requirements has led to SLA violations. In the cloud environment, it is essential to focus on several factors during the task-dependent scientific application execution, including minimizing the response time, the profit maximization, and the energy consumption minimization. The workflow scheduling model often meets the difficulty in controlling the dominance among the multi-objectives in the cloud environment. Even though several research works have focused on providing non-dominated solutions, it leads to higher computation time due to its increased comparisons over the multiple objectives. During workflow scheduling, finding the optimal solution in a non-preemptive manner is a challenging task when several parent tasks are having the same priority and different workloads in a workflow application. For this context, selecting the earliest-free VM, which

completes the execution of any of the parent tasks for its succeeding task, is inefficient. It is because, due to the various workloads, still there are several parent tasks in the running state, which increases the resource wastage and energy consumption until initiating the execution of its succeeding task in the cloud. Hence, it is necessary to effectively manage both the workflow tasks and resources while scheduling on the cloud infrastructure. To ensure the minimized energy consumption in the cloud, the dynamic consolidation of resources has great attention under the consideration of overloaded and underloaded hosts, migratable VM selection, and VM placement after migration.

In the cloud environment, scheduling similar workloads of the tasks in the same PM tends to reduce the performance due to the necessity of similar requirements during execution. Also, the dynamic variation in the resource demand has led to performance degradation concerning increased response time and computational energy. Hence, it is crucial to focus on VM migration and active resource reduction. In this chapter, we propose an energy-efficient algorithm for workflow task scheduling in clouds. Which effectively maps workflow task to VMs and consolidate overloaded and underloaded VMs using water wave-based optimization algorithm.

6.3 System Models

The proposed algorithm in this chapter manages the scheduling of the workflow tasks in the cloud environment. It aims to reduce energy consumption, improve resource utilization and minimize the number of VM migrations. To model such a capable approach, we introduced system models considered in this work. We recommend various system models such as the cloud data center, application, and energy model.

6.3.1 Data center Model

We considered that data centers offers a set of m heterogeneous hosts H $H = \{h_1, h_2, \ldots.., h_m\}$. Every host is described with various types of computing resources like CPUs, memory size, bandwidth, storage, and network capacity. These hosts or PMs are virtualized into several VMs to handle numerous user requests concurrently. The

end-users of the cloud are enabled to submit requests for the provisioning requested number of VMs. Usually, user requests are measured in million instructions (MI). Further, virtual machines are characterized by their computing efficiency in MIPS (Million Instructions Per Second), bandwidth (Bw), etc. Therefore, a cloud scheduler has different capabilities in selecting a reasonable VM to provision the user requests by satisfying the workflow constraints. Note that this chapter uses notations host and PM interchangeably to represent a cloud server.

6.3.2 Application Model

Usually, workflow applications are modeled as a DAG. A workflow W with a set of n precedence constraint tasks T_W is represents with a DAG $W = (T_w, C)$ where $T_W = \{t_1, t_2, \ldots . t_n\}$ and C is the set of communication edges $(c_{ij} \in C, 1 \leq i, j \geq n \quad and \quad i \neq j)$ that represents the dependencies between tasks. A directed arc or edge c_{ij} $(i, j \in C \quad and \quad i \neq j)$ between the tasks t_i and t_j represents the dependency, where t_i is the parent of t_j. A task having no parent(s) is an entry task t_{entry} and A task without successor(s) is an exit task t_{exit}. Any task t_i is bring on to execution when it provisioned requested computing resources. If a task t_i finishes execution then its output data transfers to all of its successors. The data (in MB) communicated from parent task t_i to its succesor task t_j has considered as the cost of edge c_{ij} i.e., $w(c_{ij})$. We express the total running time of W as makespan T_M. The communication cost between any two precedence constraint tasks t_i and t_j is represented as $T(t_{ij})$ and is calculated as in equation (6.1).

$$T(t_{ij}) = \frac{w(c_{ij})}{Bw} \tag{6.1}$$

where Bw is the bandwidth and $w(c_{ij})$ is the tranffered data between t_i and t_j. The effective execution cost of any task t_i on a specific vm is computed using equation (6.2).

$$T(t_i, vm_k) = \frac{w_i}{vm^k_{mips}} + T(t_{ij}) \tag{6.2}$$

where $T(t_i, vm_k)$ is the effective computing time of task t_i on k^{th} virtual machine vm_k, w_i is the computation cost of task t_i and vm^k_{mips} is the computing capacity of vm_k. For any task t_i a mean computing time is calculated using equation (6.3) and is the average of execution

times of all tasks on a set of virtual machines.

$$\overline{T}(t_i) = \sum_{k=1}^{n} \frac{T(t_i, vm_k)}{n} \tag{6.3}$$

we calculated the earliest start time EST and earliest finish time EFT of task t_i as follows.

$$EST(t_i) = \begin{cases} 0 & \text{if } t_i = t_{entry} \\ max_{t_p \in parent(t_i)} EFT(t_p) & \text{otherwise} \end{cases} \tag{6.4}$$

$$EFT(t_i) = EST(t_i) + T(t_i, vm_k) \tag{6.5}$$

The completion time of the exit task i.e., $EFT(t_{exit})$ is the minimum makespan of the workflow W $minT_M = EFT(t_{exit})$.

6.3.3 Energy Model

The energy dissipation of servers in data centers is primarily because of the CPU, memory, networking devices, storage media, and other underlying circuits. Among these, the CPU dominates energy consumption. The power consumption of a PM in a cloud data center can be precisely specified by a linear relationship with the CPU utilization even by applying DVFS [127]. An active physical server but with zero loads utilizes 50% to 70% of the power used by the physical server running at maximum load [23]. Hence, A linear power model proposed in [145] is used in this work. Therefore, the power utilization of a cloud server P_{util} is calculated using the following equation (6.6).

$$P_{util} = P_{CPU_{idle}} + (P_{CPU_{full}} - P_{CPU_{idle}}) * CPU_{util} \tag{6.6}$$

where $P_{CPU_{full}}$ and $P_{CPU_{idle}}$ are CPU power consumptions at loads 100% and 0% respectively and CPU_{util} is the utilization of the CPU. The energy consumption represents the product of power consumption and time. Therefore, the energy utilization is calculated when the machine is running using equation (6.7).

$$E = P_{util} * t \tag{6.7}$$

The energy consumption of t_i executing on vm_k is measured using equation (6.8).

$$E(t_i) = P_{vm_k} * T(t_i, vm_i) \tag{6.8}$$

where P_{vm_k} power consumption of vm_k and $T(t_i, vm_i)$ is the effective run time of t_i on vm_k. The total energy utilized for the workflow W execution is given by the summation of the individual tasks energy utilization in the application.

$$E(W) = \sum_{i=1}^{n} E(t_i) \tag{6.9}$$

6.3.4 Problem Specification

The proposed EASVMC algorithm aims to minimize overall energy utilization, maximizing resource utilization, minimize VM migrations and maximize dormant hosts. We formulated the mathematical notations of objectives of the proposed work using the system models discussed in Sections 6.3.1 to 4.3.3, average system resource utilization specified in Equation (6.11) and the fitness function is given in Equation (6.14).

Energy: Minimize $E(W)$
Resource utilization: Maximize $RU_{avg}(H)$
VM migrations: Minimize M
Slepping hosts: Maximize P_s

6.4 Propposed Algorithm

The proposed EASVMC algorithm is capable of minimizing energy consumption by effectively applying workflow scheduling and virtual machine consolidation methods. It is a scheduling application with a bounded number of computing machines, which has been implemented in two phases: task scheduling phase for mapping the workflow tasks to the optimal VMs to minimize energy and VM consolidation phase to maximize the number of sleeping hosts to save more energy. The implementation of the proposed EASVMC algorithm discussed in detail as follows.

6.4.1 Task Scheduling

This phase requires the priorities of tasks based on execution length without disturbing the dependencies among the tasks. The highest priority is assigned to the task with the longest execution length and the task list is generated based on the decreasing priorities. The decreasing priority order shows the topological orders of tasks which preserves task dependencies. The first task t_i is selected from the task list i.e. the task with the highest priority and calculate the energy consumed on all available VMs using Equation (6.8). The VM, that consumes the least energy is marked as optimal VM vm_k^{opt} for the task t_i and task t_i is mapped to $vm_k^{opt_i}$. The entire procedure for VM selection for the workflow tasks is shown in Algorithm 6.

Algorithm 6 VM Scheduling

```
1:  procedure VM SCHEDULING(W, H, VM)
2:      Compute the mean computation costs of each task using equation (6.3)
3:      Assign tasks priorities based on computational costs
4:      Prepare a task list for scheduling by decreasing tasks priority order
5:      while there are unscheduled tasks in the list do
6:          Select the highest priority task i.e. first task from the task list
7:          for each vm_k in the set of VMs do
8:              Calculate energy consumption of task t_i on vm_k i.e. E(t_i, vm_k)
9:              Map task t_i to the vm_k that minimizes the energy of task t_i
10:         end for
11:     end while
12: end procedure
```

For each task, Algorithm 6 needs to traverse on the set of VMs to get the optimal VM which will take minimum energy to execute. By considering n number of tasks in the workflow W, there are n mappings for resources, one for every task, and for each task, we need to traverse on its parent tasks to get the communication time which can be done in $O(n)$ time. Therefore, the overall time complexity of the VM scheduling algorithms is $O(n^2v)$, where n is the number of tasks and v is the number of available VMs.

6.4.2 VM Consolidation

This section discusses the VM consolidation problem to maximize sleeping hosts to save energy by improving resource usage of active servers (PMs). The VM Consolidation in

cloud data centers is a strict NP-Hard problem [19]. We adopt a new meta-heuristic that is inspired by shallow water wave models called water wave optimization [128] for VM consolidation (WWO-based VM consolidation). It was tested by the CEC2014 benchmark problem and verified that the WWO approach outperforms the state-of-the-art evolutionary approaches namely IWO, BBO, and GSO. Nowadays, the WWO algorithm and its improved versions have been adopting to solve different optimization problems for example flow shop scheduling [146], [147] [148], VM consolidation in cloud data centers [24], satellite system design [149] etc. The original WWO is proposed to solve continuous optimization problems. The VM consolidation is a combinatorial problem hence we modified parameters of WWO accordingly to solve discrete optimization problems. The WWO algorithm is efficient in searching for near-optimal solutions in multi-dimensional global optimization problems [24]. The proposed technique dynamically migrates the VMs to improve the host PM resource utilization and reduce energy utilization. It also switches off idle host machines after migrating all VMs from it to a target host to save more energy.

We considered a bounded number of host machines (PMs) and virtual machines in this work to carry out the VM consolidation problem. Based on the resource (CPU) utilization the proposed algorithm categorizes the PMs into three groups: over-utilized PMs H_{over}, normal-utilized PMs H_{normal}, and underutilized PMs H_{under}. For any single host, the percentage of resource utilization (RU) is calculated using equation (6.10) and the overall system resource utilization of active hosts using equation (6.11).

$$\%RU_{h_i} = \sum_{j=1}^{k} \frac{vm_{mips}^{i,j}}{h_{mips}^{i} * k} * 100 \tag{6.10}$$

where $RU(h_i)$ is the resource utilization of the i^{th} physical machine, $vm_{mips}^{i,j}$ is the MIPS acquired by j^{th} vm on host h_i, h_{mips}^{i} is the MIPS of the i^{th} physical machine and k is the total of VMs on host h_i.

$$RU_{avg}(H) = \frac{\sum_{i=1}^{n} RU(h_i)}{n} \tag{6.11}$$

where $RU_{avg}(H)$ is the average resource utilization for n number of hosts. We used double (upper and lower) threshold values to categorize PMs based on the percentage of the host resource utilization. All the hosts with resource utilization less than the lower threshold L_{th} are H_{under}, the hosts with resource utilization over the upper threshold U_{th} are H_{over} and the rest of the hosts are H_{normal}. The upper threshold and lower threshold for a host resource

utilization are set to 0.9 and 0.5 [150] respectively. The pseudo-code for the process of identifying the type of host is shown in Algorithm 7.

Algorithm 7 Host Categorization

```
1:  procedure HOST CATEGORIZATION(H)
2:      for each host h_i do
3:          Calculate host utilization RU(h_i) using Equation (6.10)
4:          if RU(h_i) < L_th then
5:              Add host h_i to H_under
6:          end if
7:          if RU(h_i) > U_th then
8:              Add host h_i to H_over
9:          else
10:             Add host h_i to H_normal
11:         end if
12:     end for
13: end procedure
```

In terms of VM consolidation in cloud data centers, any PM is a potential source host or destination host. The proposed VM consolidation algorithm prepares a set of three-element tuples T $t = (h_{source}, vm, h_{dest})$ where h_{source} is the source host to migrate the one or more VMs, vm is the specific VM selected on h_{source} for migration and the h_{dest} is the target host to accommodate the VM(s) from h_{source}. While making tuples the algorithm ensures two constraints to restrict generating too many tuples, which improves the time complexity of the algorithm without affecting its performance. The first constraint is the source vm for migration should be not from the normally loaded hosts.

$$h_{source} \in H_{under} \vee h_{source} \in H_{over} \tag{6.12}$$

where H_{under} and H_{over} are set of underloaded and overloaded hosts respectively. The second constraint to restrict the size of the tuple set further is that the no overloaded host becomes the destination host.

$$h_{dest} \notin H_{over} \tag{6.13}$$

The proposed algorithm aims to maximize the number of sleeping hosts by hosting all the VMs on the active hosts without compromising their performance. To accomplish this task, the VM consolidation technique is enforced. Which employ live VM migration to transfer VM(s) between physical hosts to improve the resource utilization and energy efficiency in the data center.

6.4.3 Water Wave based VM Consolidation

The shallow water wave-inspired WWO approach is modeled to solve the optimization problems. With a maximizing objective function f and solution space X (which is comparable to the seabed area), the fitness of a point $\mathbf{x}$ ($\mathbf{x} \in X$) is inversely proportional to the seabed area depth. In simple words, the fitness of any point $\mathbf{x}$ $f(\mathbf{x})$ is higher if the point $\mathbf{x}$ is close to the still waters. The variety of shapes of a wave is shown in Figure 5.1. The WWO algorithm maintains a set of solutions (population) like other evolutionary algorithms (EAs). Algorithm 8 gives the pseudo-code for the random population (solutions) generation.

Algorithm 8 Population Generation

```
1:  procedure POPULATION GENERATION(H)
2:      H_source ← H_under ∪ H_over
3:      H_target ← H_normal
4:      for each host h_i ∈ H_source do
5:          for each vm_k on h_i do
6:              for each host h_j ∈ H_target do
7:                  Add this tuple (h_i, vm_k, h_j) in a possible migration set
8:              end for
9:          end for
10:     end for
11:     while Polulation size is less than desired do
12:         x ← EmptyWave
13:         Indices ← generate a random permutation of size of all possible set
14:         for each indices_i ∈ Indices do
15:             Add tuple at index as indices_i from possible migration set to x
16:         end for
17:         Add this x to population set P
18:     end while
19: end procedure
```

The population set P for WWO is generated by the set of possible migration of VMs from over-utilized and underutilized hosts to normal-utilized hosts. The migration from over-utilized hosts to normally utilized hosts balances the load and from underutilized hosts to normally utilized hosts free the hosts then they can be turned off. Each solution is a migration plan and we represented it as a wave $\mathbf{x}$ with three parameters amplitude (height) h, wavelength λ and a tuple t. After the generation of population set P, the search for the best migration plan starts. This search process of the WWO algorithm in the solution

space is modeled as the propagation, refraction, and breaking of the waves. Initially, the wavelength and height of all waves in the population set are assigned as 0.5 and h_{max} respectively, and the fitness of each wave **x** is calculated using equation (6.14).

$$f(\mathbf{x}) = P_s^{\gamma} + 1/(\varepsilon + M) \tag{6.14}$$

where, P_s is the number of switched off hosts (PMs), M is the number of migrations, γ is a parameter that defines the relative importance of P_s, and ε is a small value to avoid division with zero when there are no migrations.

Propagation: During the propagation process, a wave needs to be generated exactly once. If the wavelength λ of current wave **x** is less than a randomly generated value r then **x** generates exactly one new wave **x'**. A new tuple t is added to the offspring wave **x'**, replace the existing tuple with a new tuple, or removing the existing tuple is performed based on two probabilities r_1 and r_2. These r_1 and r_2 are lower and upper probabilities respectively and are based on series of preliminary experiments to improve the performance of the propagation. If the fitness of **x'** is greater than the fitness of **x** then replace **x** by **x'** and set height of **x** to h_{max}. Otherwise, the wave **x** remains but reduces the height of **x** by one. For every generation, the wavelength is updated as follows:

$$\lambda = \lambda \cdot \alpha^{-(f(\mathbf{x})-f_{min}+\varepsilon)/(f_{max}-f_{min}+\varepsilon)} \tag{6.15}$$

where, $f(\mathbf{x})$ is fitness function, α is a wavelength reduction coefficient, f_{min} and f_{max} are minimum and maximum fitness of the current solution respectively, and to avoid division with zero in case of no migrations a small values ε is used. When fitness is higher the wavelength is less for any wave, hence equation (6.15) ensures propagation in smaller areas.

Refraction: In the theory of a water wave, a wave is refracted when its ray is not perpendicular to its isobath. A wave refraction is performed on the wave **x** when its heigh becomes zero i.e., $\mathbf{x}.h = 0$. Suppose **x*** is the global best wave so far and the height of a wave **x** becomes zero then the wave **x** is replaced by new wave **x'**. The position of the new wave **x'** is at the midpoint between the current wave **x** and the global best wave **x***. In the

context of VM consolidation, every solution is a migration having some tuples representing the solution. When a solution is poor i.e. height h vanishes then add some tuples or replace the existing tuples in the current solution with some tuples from the global best solution. And for every new solution height must be set to h_{max} and then update its wavelength λ' as in equation (6.16).

$$\lambda' = \lambda \cdot \frac{f(\mathbf{x})}{f(\mathbf{x'})} \tag{6.16}$$

where $f(\mathbf{x})$ and $f(\mathbf{x'})$ are the fitnesses of old and new solutions respectively.

Breaking: Breaking operation performed to find the new best solution if any which will be carried by conducting a local search. It is only performed on the wave that gives a better solution than the global best solution. Suppose a newly generated solution **x** has better fitness than the global best solution **x*** i.e. $f(\mathbf{x}) > f(\mathbf{x^*})$ then **x** becomes the new global best solution i.e. $\mathbf{x^*} = \mathbf{x}$. Conduct a local search around the new global best solution for a better solution than the current **x***. If it finds a better solution then the new solution **x'** becomes the new global best $\mathbf{x^*} = \mathbf{x'}$, otherwise **x*** remains the global best solution.

The breaking operation can be performed for a better solution in the following way: After the propagation, if a solution **x** found to be better than the **x***, then we look for a better solution in the neighborhood of **x**. For a fixed number of times let say k, select a few tuples (say 5) from the current best solution. Then for each of the selected tuple replaces the tuple with a tuple from T excluding the tuples from **x**. This can produce series of solitary waves, If the solitary wave found to be a better solution **x'** than **x*** then $\mathbf{x^*} = \mathbf{x'}$. The minimum and maximum values for k are 1 and 12 respectively i.e the value of k is a predefined number [128].

Algorithm 9 contains the pseudo-code for the VM consolidation phase. The main objectives of this algorithm are dynamic VM migrations for load balancing and free some hosts to switch off, hence saving energy. It also maximizes the resource utilization of active hosts. In the first two steps, it calls two algorithms a host categorization and population generation. Further, it performs propagation, breaking, and refractions operations for finding the best migration plan.

The time complexity of the VM Consolidation phase is $O(h^2v + pw(1+mw))$. where h is the number of hosts, v is the number of virtual machines, p is the population size, w is

Algorithm 9 VM Consolidation

```
1: procedure VM CONSOLIDATION(H, VM)
2:     HOST CATEGORIZATION(H)
3:     POPULATION GENERATION(P)
4:     while stopping criteria not reached do
5:         for each wave x ∈ P do
6:             x' ← Perform propagation
7:             if f(x') > f(x) and f(x') > f(x*) then
8:                 Perform breaking operation on x'
9:                 Update x, x* ← x'
10:            else decrement height of x by 1
11:            end if
12:            if height of x vanishes then
13:                Perform refraction operation on x to a new x'
14:                Update Wavelength using equation (6.15)
15:            end if
16:        end for
17:    end whilereturn Migration Plan
18: end procedure
```

TABLE 6.1: The EASVMC algorithm summary

	Maximize resource utilization (%)	Reduce energy	VM Migrations	Time complexity
VM Scheduling	-	Yes	-	$O(n^2v)$
Host categorization	-	-	-	$O(h)$
Population Generation	-	-	-	$O(h^2v+pw)$
VM Consolidation	Yes	Yes	Yes	$O(h^2v+pw(1+mw))$

the wave size and m is the number of iterations which WWO will perform. To prepare the all possible migration tuples algorithm takes $O(h^2v)$ time, for population generations $O(pw)$ time and to perform the wave operations such as propagation, refraction, and breaking $O(mpw^2)$ time. Table 6.1 summarizes the functions and time complexity of each sub-algorithm of EASVMC.

TABLE 6.2: Workflow benchmark setup

Workflow	Number of tasks
Montage	50, 75, 100, 125, 150 and 1000
CyberShake	90, 120, 150, 180, 210 and 1000
LIGO	74, 100, 124, 150, 174 and 1000
SIPHT	100, 150, 200, 250, 300 and 1000
Epigenomics	24, 46, 64, 100, 125 and 997

TABLE 6.3: VM Instance Specifications

VM type	Memory (in GB)	Number of cores	Computing Capacity (MIPS)
Micro	1	1	500
Small	2	1	1000
Medium	4	2	2000
Large	8	2	2500

6.5 Performance Evaluation

We discuss the efficiency of our proposed EASVMC approach in this section. we evaluated the performance of the EASVMC through a sequence of simulation experiments for energy consumption, resource utilization, VM migrations, and the number of switched-off hosts.

6.5.1 Experimental Settings

We used the toolkit called WorkflowSim to simulate the cloud environment, which enables users to simulate the Infrastructure-as-a-Service (IaaS) cloud. The IaaS cloud provides a virtualized platform for scheduling workflow applications. We choose five different workflows such as CyberShake, Montage, SIPHT, LIGO, and Epigenomics from various scientific areas to evaluate the performance of our proposed EASVMC algorithm. The topological structures of these five workflows are depicted in Figures 1.3 to 1.7. For every workflow, we consider five different workloads in the simulation experiments Table 6.2. Cybershake is a memory-intensive and data-intensive earthquake science application. Montage is an astrophotography application and is characterized as an I/O-intensive astronomy application.

SIPHT is the sRNA Identification Protocol using High throughput Technology (SIPHT) program that is used in bioinformatics applications to empower high-throughput, kingdom-wide prediction, and functional annotation of bacterial sRNA-encoding genes [15]. Epigenomics is the CPU-intensive workflow [14], which is the bioinformatics application that enables the automation of several genome sequencing operations. The Laser Interferometer Gravitational-Wave Observatory (LIGO) is a driving application to detect gravitational waves developed by violent events in the universe. Usually, very largescale datasets are used in scientific workflow applications. Each scientific workflow has its data and computing power requirements and their complete characterization presented by Juve et al. [17]. We have considered four different VM instances in CDC, each VM has its computing resources as given in Table 6.3 and is self-defined. Each VM type has virtually unlimited instances and a constant Bw among VM instances. Along with ε and γ, the other control parameters used in WWO such as maximum wave height, wavelength reduction coefficient α, the maximum number of breaking directions k_{max}, wave propagation probabilities r_1 and r_2 for getting good solution are tabulated in Table 6.4. All these recommended values for these parameters are based on series of experiments to obtain an optimal solution. The smaller values for h_{max} improves the solution diversity by frequently replacing the solution with new solutions and the large value specifies the more life span for the solution. The α values define the size of the area of solution exploration, small values cause intensive explorations.

TABLE 6.4: WWO Parameters.

γ	ε	r	r_1	r_2	α	λ	k_{max}
5	0.00001	(0, 1)	.28	.68	(1.001, 1.01)	0.5	12

6.5.2 Self-Comparison Experiments

We conduct self-comparison experiments of energy consumption and resource utilization in this section. To show the energy-saving and resource utilization effect of our sub-algorithms, we use the VM scheduling algorithms as a baseline. It is noted that the VM consolidation algorithm includes two sub-algorithms, i.e., host categorization algorithm and population generation algorithm. The simulation results of self-comparison for energy-saving and maximizing resource utilization are given in Figure 6.1, of five scientific workflows. Based on the VM scheduling algorithm, we can see from Figure 6.1 that the VM Consolidation

algorithm indeed saves more energy and maximizing resource utilization. This is because the VM consolidation algorithm always finds underutilized hosts and switch-off them after migrating its VMs to a suitable target host and migrate VMs from the overloaded hosts for load balancing. Hence it can save more energy and maximize resource utilization.

6.5.3 Comparison Experiments with Others

First, we conducted simulation experiments to expose the capability of the VM consolidation EASVMC algorithm in reducing energy consumption and maximizing resource utilization. We perform three well-known VM consolidation algorithms to compare with our VM consolidation algorithm. These algorithms operate the CPU by keeping its load between thresholds. If a host belongs to an underutilized or over-utilized category then the VMs running on such host are migrated to a suitable target host for load balancing. The three reference algorithms are one heuristic algorithm for the dynamic reallocation of VMs in [127] which dynamically sets utilization threshold to depend on the Median Absolute Deviation (MAD), and two meta-heuristics ACS-VMC [19], and WWO-VMC [24]. We used parallel workloads in [132] for the simulation experiments. The simulation experimental results for the four algorithms in energy consumption depicted in Figure 6.2 for different workloads. Among these VM consolidation policies, our VM consolidation algorithm has the least energy consumption, the MAD performed worst in energy consumption scheduling, and the other two algorithms have medium energy consumption.

Then, we run our EASVMC algorithm for energy consumption, resource utilization, number of *vm* migrations, and number of switched-off hosts. To reveal the strength of the proposed EASVMC scheduling approach, we also perform three popular scheduling approaches HEFT [92], EES (Enhanced Energy-efficient Scheduling) [63] and PESVMC [77] for different objectives. The HEFT and EES (approaches are prominent list-based heuristics for scheduling workflow applications. The power-efficient scheduler with the virtual machine consolidation (PESVMC) approach efficiently reduces the energy consumption by switching off the underutilized hosts by performing VM migrations to a suitable destination host. The PESVMC algorithm identifies the underutilized and over-utilized host machines by using lower and upper utilization threshold values and switch off underutilized host machines by migrating its VMs to normally utilized hosts.

The simulation experimental results for energy consumption and resource utilization on five different real-world scientific workflows are depicted in Figures 6.3 to 6.8.

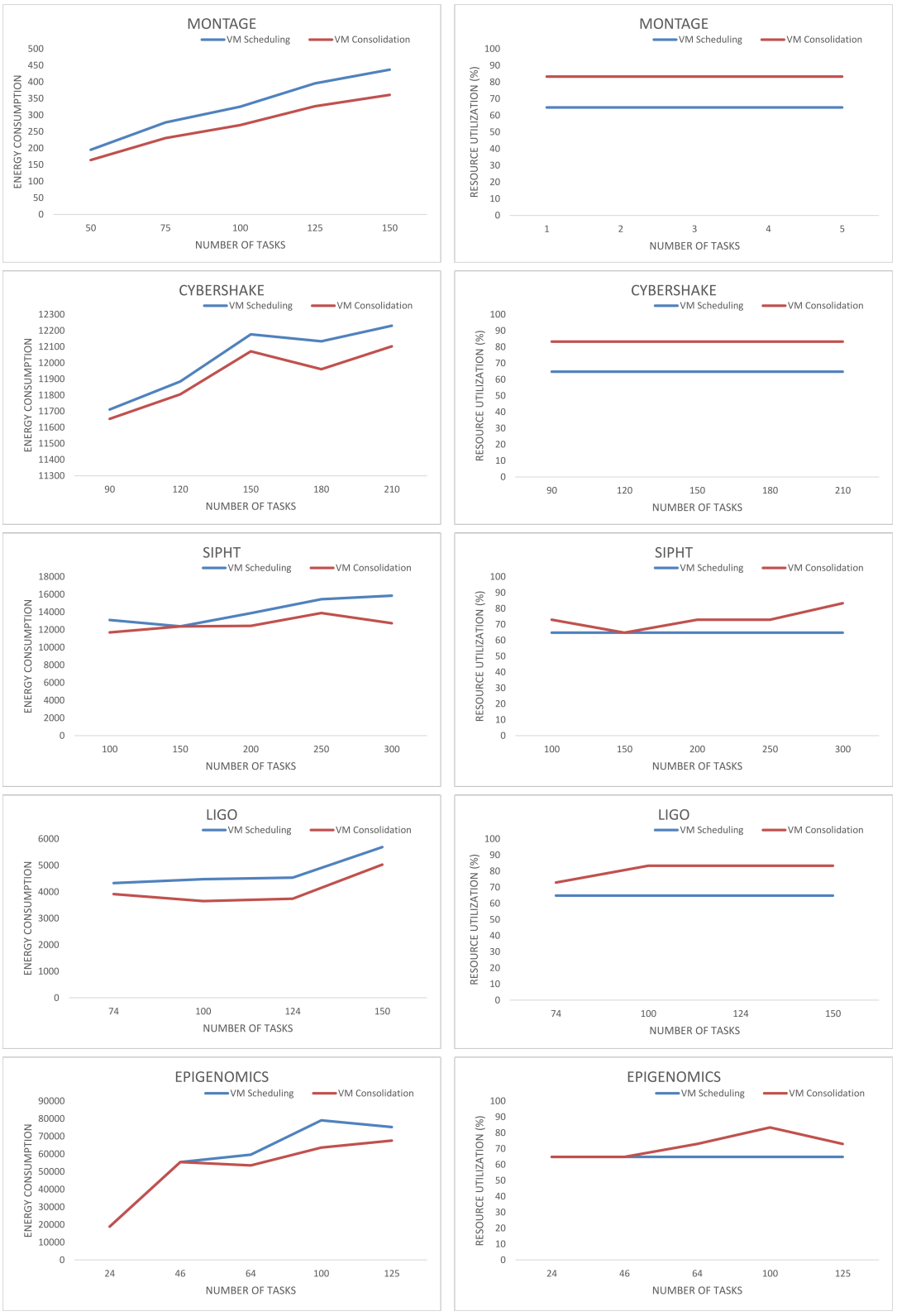

FIGURE 6.1: Self-comparison experiments in energy consumption and resources utilization.

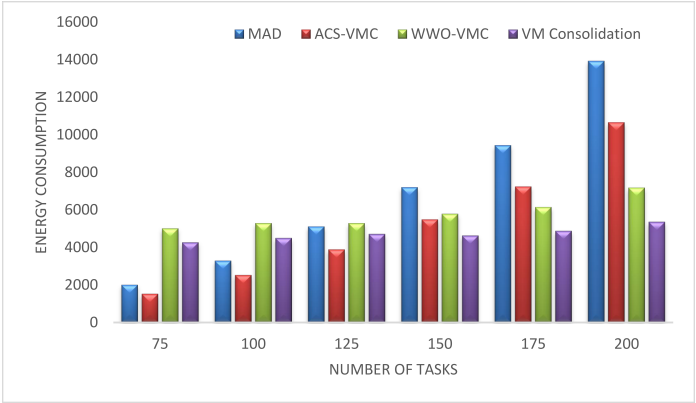

FIGURE 6.2: Comparison experiments of VM consolidation algorithm in energy consumption.

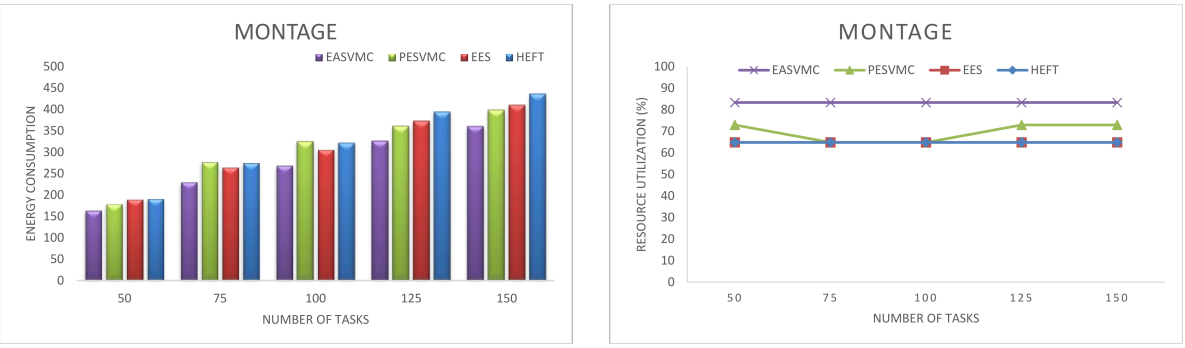

FIGURE 6.3: Energy consumption and Resources utilization on Montage Workflow.

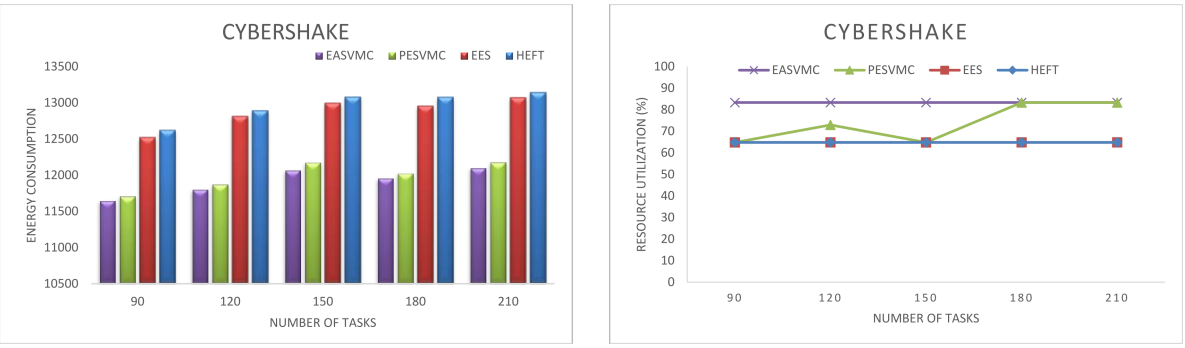

FIGURE 6.4: Energy consumption and Resources utilization on CyberShake Workflow.

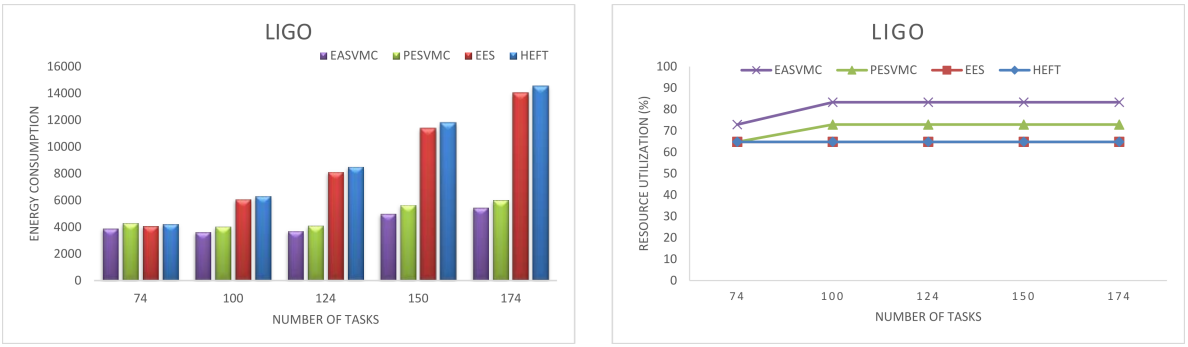

FIGURE 6.5: Energy consumption and Resources utilization on LIGO Workflow.

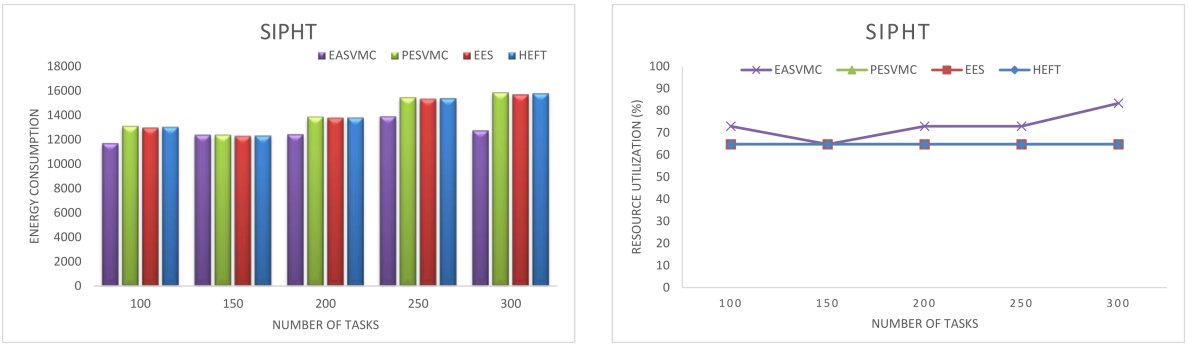

FIGURE 6.6: Energy consumption and Resources utilization on Sipht Workflow.

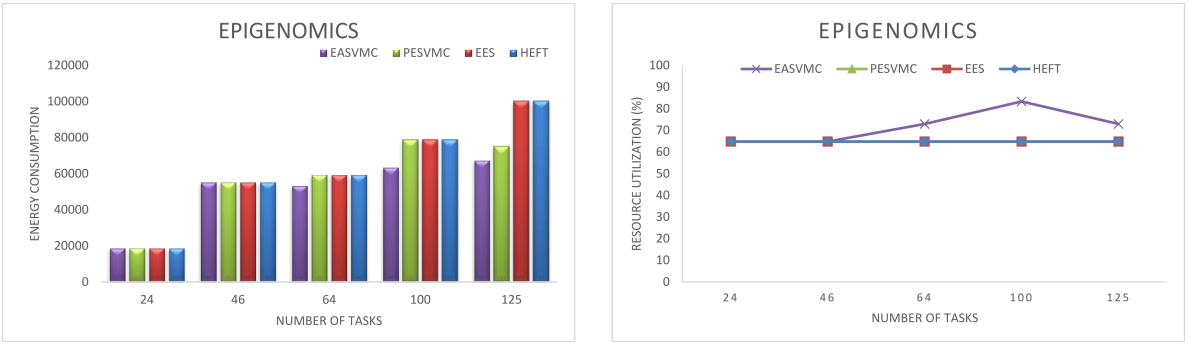

FIGURE 6.7: Energy consumption and Resources utilization on Epigenomics Workflow.

TABLE 6.5: Number of VM migrations and Switched off hosts

Workflow	Number of migrations				Number of switched off hosts			
	HEFT	EES	PESVMC	EASVMC	HEFT	EES	PESVMC	EASVMC
Montage	0	0	7	8	0	0	1	2
	0	0	6	8	0	0	0	2
	0	0	6	8	0	0	0	2
	0	0	6	8	0	0	1	2
	0	0	7	8	0	0	1	2
Epigenomics	0	0	0	0	0	0	0	0
	0	0	1	1	0	0	0	0
	0	0	3	6	0	0	0	1
	0	0	6	7	0	0	0	2
	0	0	4	4	0	0	0	1
SIPHT	0	0	3	6	0	0	0	1
	0	0	2	2	0	0	0	0
	0	0	5	5	0	0	0	1
	0	0	6	6	0	0	0	1
	0	0	6	7	0	0	0	2
LIGO	0	0	3	6	0	0	0	1
	0	0	6	8	0	0	1	2
	0	0	6	9	0	0	1	2
	0	0	6	10	0	0	1	2
	0	0	6	9	0	0	1	2
CyberShake	0	0	6	8	0	0	0	2
	0	0	6	8	0	0	1	2
	0	0	7	7	0	0	0	2
	0	0	9	7	0	0	2	2
	0	0	7	8	0	0	2	2

We conducted several simulations runs for energy consumption and resource utilization on each workflow for different workloads. Our proposed discrete WWO algorithm is an efficient global optimization algorithm that is successful in selecting the optimal target host machines to place the VMs in an energy-efficient way. Hence it saved significant energy compared with other works with different workflows. Further, the resource utilization of the EASVMC algorithm maximized compared with other works. In Figures, 6.3-6.7 the resource utilization results for HEFT and EES are overlapped. The PESVMC algorithms failed to identify the idle hosts to switch off on both Sipht and Epigenomics workloads in Figures 6.6 and 6.7 respectively. However, consumed less percentage of energy compared with HEFT and EES. This is because of its efficiency in selecting energy-efficient resources to execute tasks. The average resource utilization for the four algorithms depicted in Figure 6.8. Results of HEFT and EES are overlapped and the EASVMC outperformed all other three techniques.

The simulation results for the number of *vm* migrations and the number of switched-off hosts are given in Table 6.5. The HEFT and EES algorithms have not implemented VM consolidation techniques hence it has zero entries in respective columns. Compared with the PESVMC algorithm our proposed EASVMC algorithm efficient in identifying and switching off idle hosts and reducing overall *vm* migrations. Further, we evaluated our algorithm for large workloads of the five scientific workflows and the results are tabulated in Table 6.6 for energy consumption, resource utilization, number of *vm* migrations, and number of switched-off hosts. In all the cases our proposed EASVMC algorithm surpasses other algorithms. The efficiency of the EASVMC in average energy-saving against the other three algorithms is given in Table 8. In summary, the experimental results reveal that our EASVMC algorithm shows a significant performance advantage over the related well-known techniques in maximizing resource utilization and energy saving irrespective of diverse workflow structures. This is because of the efficiency of VM scheduling algorithms in selecting the resources and the proposed WWO based VM consolidation algorithm minimizes the number of active hosts and maximizing resource utilization.

TABLE 6.6: The EASVMC performance for large workloads

Workflow	Energy consumption				Resource utilization (%)			
	HEFT	EES	PESVMC	EASVMC	HEFT	EES	PESVMC	EASVMC
LIGO-1000	90641.24	79231.32071	26773.19	23865.87	64.81	64.81	83.33	83.33
SIPHT-1000	21831.51	20877.47301	21994.34	20497.15	64.81	64.81	83.33	83.33
Montage-1000	2646.25	2527.16875	2474.61	2160.55	64.81	64.81	72.92	83.33
CyberShake-1000	20241.5	20160.534	15889.46	15518.38	64.81	64.81	72.92	83.33
Epigenomics-997	804587.1	757921.067	284259.2	236692.8	64.81	64.81	64.81	83.33
	Number of migrations				Number of switched off hosts			
LIGO-1000	0	0	7	7	0	0	2	2
SIPHT-1000	0	0	12	7	0	0	2	2
Montage-1000	0	0	7	7	0	0	1	2
CyberShake-1000	0	0	8	7	0	0	1	2
Epigenomics-997	0	0	5	8	0	0	0	2

TABLE 6.7: Energy-savibg of the EASVMC algorithm

Workflow	Energy Saving (%)		
	vs. HEFT	vs. EES	vs. PESVMC
Montage	20	15	14
CyberShake	8.1	7.7	.8
LIGO	113	86	9.1
SIPHT	11.5	6.5	12
Epigenomics	21	14	11.5

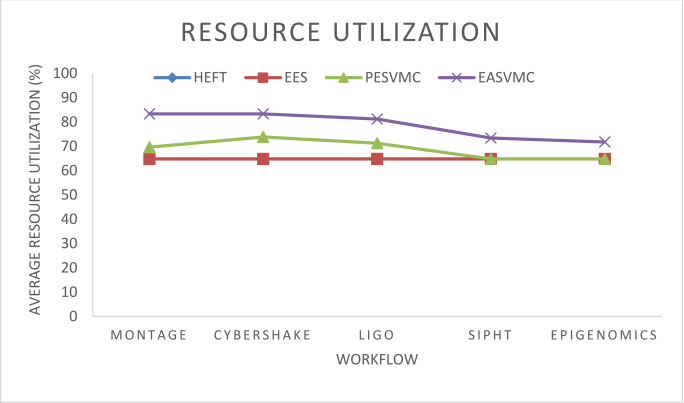

FIGURE 6.8: Average resource utilization on different Workflows.

6.6 Conclusion

This work presented the energy-aware scheduling model for the workflow applications, which maintains the trade-off between the performance and energy consumption in the cloud environment. It reviews the existing research works on energy-aware scheduling in a cloud environment. From the analysis of the conventional research works on the cloud, there is an essential need for performing dynamic and energy-efficient workflow scheduling and ensuring QoS to the end-users. Hence, the proposed approach has modeled the energy-efficient scheduling framework with discrete water wave-based VM consolidation to ensure the reduced energy consumption while assuring the quality of service. The shallow water wave theory inspired WWO approach to perform well in consolidating the VMs to save energy. The algorithm tested for different objectives such as energy consumption, resource utilization, number of *vm* migrations, and number of

switched-off hosts. We have compared the performance of our algorithm with three well-known approaches HEFT, EES, and PESVMC. The simulation results show that the overall performance of the EASVMC algorithms outperformed the other three approaches.