Chapter 2

Review of Scheduling Approaches for Workflows in Clouds

This chapter[1] *presents an extensive review of the existing energy-aware workflow scheduling algorithm in a cloud environment. We have studied and presented the state-of-the-art algorithms from different perspectives and presented their key ideas, strength, and limitations. We have provided some future directions to further improve the existing works.*

2.1 Introduction

Due to the increasing amount of data worldwide, most business organizations today are adopting cloud technology. Most consumers and businesses are using the cloud because it is convenient, adaptable, massive-scalable, and secure. Exceptionally, high availability and scalability features of cloud infrastructure attracting the end-users. Cloud computing practices have evolved the ways we use computers and have magnified the power of the internet more than ever. Due to the ever-increasing cloud applications, there is a massive increase in computing demand on data centers [24]. Gartner, Inc, a Global leading research firm predicted a 23% increase in public cloud user spending in the year 2021.

[1] This chapter is derived from: Medara, R., Singh, R.S. A Review on Energy-Aware Scheduling Techniques for Workflows in IaaS Clouds. Wireless Pers Commun (2022). https://doi.org/10.1007/s11277-022-09621-1

Consequently, data centers become unsustainable because of higher energy utilization, which results in higher energy costs and substantial carbon footprints. Global data centers use approximately 416TW (4.16 x 1014W) of electrical power, which is around 3% of the whole power-producing on the planet [26], and this consumption is expected to double every four years [27]. Consequently, data centers releasing green gases into the environment. The Global data center's CO_2 emissions are approximately equal to the CO_2 emissions of the aviation industry [28], which accounts for 2% of global human-made CO_2 emissions. The ICT (information and communications technology) services are expected to utilize 20% of global electrical power by 2025 and discharge nearly 5.5% of all carbon emissions in the world [29]. Due to the higher energy consumption of cloud data centers, the service providers experience high operating costs, resulting in increased "cost of ownership." In addition to economic damages, the greenhouse gases (GHGs) released by data centers affecting the environment, along with the current rate of CO_2 emissions of data centers, are expected to surpass airline industry emissions soon. It is predicted that the total carbon footprint of the ITC sector will account for as much as 14% of the entire world's emissions by 2040 [30].

Energy efficiency is essential for the cloud data center for two reasons 1) data center operational cost optimization and 2) to improve the environment. Optimization of energy consumed by the cloud-related infrastructure is the major research deal in recent years. The IT infrastructure is the main contributor to the energy consumption in a data center which includes servers and other IT components. The cloud application scheduling is extensively used as a productive energy conservation method. An efficient scheduling mechanism is required to 1) optimize the cloud resources 2) provide the end-users high efficiency and 3) be able to provide high-quality client service. The server's energy consumption can appreciably be reduced by consolidating cloud applications as on few servers as possible and shut down idle servers [31].

Most of the business and complex scientific applications used workflows to analyze the complex data sets and to conduct simulation experiments effectively [7]. The scheduling of workflow tasks in distributed platforms such as grids and cloud environments has been extensively considered for many years. Researchers have developed algorithms geared towards different environments: from small-scale homogeneous clusters to large-scale community grids, to the contemporary paradigm, heterogeneous, utility-based, and resource-rich cloud computing [7]. This chapter focuses on cloud computing platforms; it

surveys algorithms developed to coordinate the execution of energy-aware workflow jobs in cloud computing environments, specifically, IaaS clouds.

The rest of this chapter is organized as follows: the overview of the workflows introduced in Section 2.2. Section 2.3 discusses a cloud workflow management system and the workflow applications scheduling problem in a cloud environment. Section 2.4 provides a relatively comprehensive review of literature on the existing energy-aware workflow scheduling techniques and presents a new classification. Finally, in Section 2.5, we discussed the conclusions of the survey.

2.2 Overview of Workflow

The origin of the workflow concept and five real-world workflows from different scientific fields are introduced in Section 1.1.2. A workflow can be described as a sequence of computational tasks together with dependencies. In scientific or business applications these task dependencies perform data communication. As an example, Figure 1.1 shows a simple workflow with ten nodes. Each vertex represents a task and the edge between vertices is the task dependency. The outcome of one task becomes the input file to another task. A task has its computing characteristics such as CPU-bound, memory-bound, or I/O-intensive (or even a combination of three).

The various works considered in this chapter and the further proposed new algorithms mostly used DAG-based workflow models. Most of the works used the application model defined as follows:

A workflow application with a group of tasks $T_w = t_1, t_2, t_3, \ldots . t_n$ while their dependencies can be modeled as a DAG $W = (T_w, E)$ where E is the set of edges represents the dependencies among tasks. An edge e_{ij} is the dependency from t_i to t_j where t_i is the parent of t_j and t_j is the child of t_i. A task without a parent(s) or predecessor(s) is called entry task (t_{entry}) and a task without child(s) or successor(s) is called exit task (t_{exit}).

2.3 Workflow Scheduling in Clouds

To preserve the dependencies among tasks in workflow applications while executing on distributed environments such as clouds, we need to consider the individual task mapping to the resources and orchestrating their performance [32]. This mapping problem needs to satisfy the functional and non-functional Quality of Service (QoS) requirements. The QoS indicates the levels of performance, reliability, and service availability offered by an application and by the platform or infrastructure that hosts it [33]. Large-scale scientific workflows execution experience increase in system randomness and unpredictable workloads such as execution time, variable cost factors, etc., [34] and oscillating workloads make this problem computationally intractable [35], which makes this problem to be in the class of NP-hard problems.

The workflow application execution plan considers two steps. The first one is the resource provisioning, which is to select the suitable resources to execute the workflow tasks, the second one is the task allocation or actual scheduling phase to optimally dispatch the workflow tasks to the selected resources [7] [36]. Execution time, overall financial cost, and energy consumed by the workflow application are all affected by the selected resources; hence, the heuristics that were used in the first step are capable of selecting the number of VMs, VMs types, and when to start them and when to shut them down.

2.3.1 Cloud Workflow Architecture

The aim of this survey is to study the various energy-efficient techniques for scheduling workflow applications in a cloud environment. Note that the works in [37] [2] [38] proposed cloud system architecture for implementing workflow applications in an energy-efficient way and [7] [39] proposed a reference architecture for Cloud Workflow Management System (CWfMS) which empowers establishment, and execution of workflows. Based on these works, we offer a reference architecture for an energy-aware Cloud Workflow Management System, which is shown in Figure 2.1. The various components depicted are standard for most of CWfMSs.

User Interface enables the users to set up, modify, submit, and track their applications.

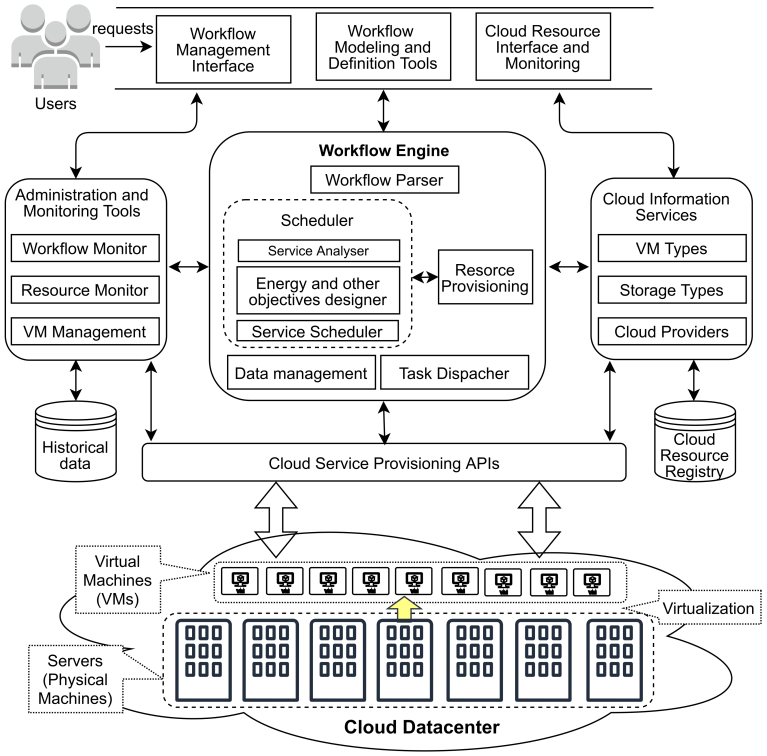

FIGURE 2.1: Workflow management architecture

Workflow Engine is the essential component of the eCWfMS and is accountable for the execution of workflow application for this it can have various sub-components such as

a) Workflow Parser is responsible for converting high-level workflow descriptions such as XML to internal specifications such as objects, tasks, parameters, and dependencies which are accessed by the scheduler component. **b) Cloud Scheduler** works with the **c) resource provisioning** modules for planning the execution of the actual workflow algorithm. The overall performance of the system, total financial cost, and energy consumption are depends on the efficiency of the scheduler. For the cost and energy efficient workflow execution, it needs to interact with different components such as:

Service analyzer- to analyze the service requirements of an application.

Energy and other objectives designer to minimize the energy consumption and some other optimization objectives such as execution cost, system reliability, etc. The cost optimizer includes *Pricing model* - is to calculate the execution cost of the application in accordance with the pricing model of the VMs.

Service scheduler- to allocate the request to computing resources such as leased VMs and takes decision when to start or stop service of VMs And often interacts with the *VM Manager* to update the status of VMs and as well as to provision new VMs across the physical machines of cloud infrastructure.

Administration and monitoring tools include monitoring modules for tracking the status and performance of workflow tasks, continuously and dynamically and enables leased resource management such as VMs. The collected data by these can be stored in historical databases, which can be useful for performance predictions of the system.

Cloud Information Service (CIS) provides information about different cloud service providers, their resource types such as VM types including pricing models. For example, Amazon EC2 giving different kinds of instances (a virtual server) for various instance families such as small, medium, large and xlarge (extra-large). The computing capabilities such as CPU computing power, memory, and storage depends on the instance type.

Cloud Service Provisioning APIs are the gateway to enable cloud services directly or indirectly for user requests. The APIs allow provision or de-provision of the computing resources on-demand. It also tracks and maintains the status of the provisioned resources including network configurations. These APIs can enhance the cloud experience by allowing the integration of applications and different workloads with cloud services and enable cross-cloud compatibility, which allows the tenants to access resources of secondary cloud providers as well, along with primary ones. The leading interoperability cloud API platform creators are Apache (Citrix) CloudStack, Amazon Web Services API and Eucalyptus, Google Compute Engine, Simple Cloud, OpenStack API, and VMware vCloud API.

2.4 Survey

We have done an extensive study on various techniques [40] to [65] developed for "Energy-Aware Workflow Scheduling in Cloud Computing Environment." Our

investigation focused on different perspectives such as algorithm evaluation environment, workflows used, application model, scheduling model, resource model, energy optimization techniques, and comparison of energy-aware methods.

2.4.1 Workflows used and evaluation environment

This section presents whether the algorithms were evaluated on a real cloud or simulation environment and also gives whether the algorithms were evaluate using the real-world scientific workflows discussed in section 1.1.2, any other specific application, or randomly generated. The summary of the algorithm evaluation environment and used workflow(s) of the surveyed algorithms summarized in Table 2.1.

Various challenging issues need to be overcome in the cloud computing field such as load balancing, and energy constraints, task scheduling, computation offloading, cost constraints, and security issues. It is costly to set up the live cloud for individuals and small companies to test their cloud applications; hence, most of the researchers selected simulation environments to test their application performance. The simulators are that they can provide users with practical feedback when designing real-world systems. The Cloud-Simulators are a possible way to review the cloud components' behavior and performance against distinctive situations and workloads. These can assist an analyst to model numerous sets of cloud applications by creating data centers, VMs, and other utilities which can be combined to configure it, so building it is very simple to analyze. Khalil, Khaled M., et al. [66] discussed the typical architecture of cloud-simulators in evaluation studies of the different types. "Cloud-Simulators."

2.4.2 Application Model

Cloud workflow application models are presented in this section which contains all instances of workflow processes. The scheduling techniques discussed in this survey differ in their workload scheduling ability. The algorithms able to schedule, a single workflow, or multiple workflows simultaneously.

The algorithms in this survey were used to schedule different workflow models such as single, multiple, and ensemble. The type of workflow used in the surveyed algorithms is summarized in Table 2.2.

Single workflow In this category, algorithms are designed to optimize the schedule of a single workflow. This model can be used in different parallel environments such as grid computing, cluster computing, and as well as is well suited for cloud computing. The scheduling techniques can optimize makespan, cost, and energy by meeting QoS requirements.

Workflow ensembles The algorithms in [62] use workflow ensembles. Large-scale applications, in particular, the inter-related scientific workflow applications are usually grouped into ensembles. Usually, ensemble workflows have a similar structure but have different input data, task sizes, and numbers of tasks in workflows. Along with workflow parameters, the priorities of workflows in an ensemble may differ the number of workflow instances in an ensemble known in advance to the scheduler.

Multiple workflows Unlike workflow ensembles, here the workflows need not be interrelated and might vary in parameters. The scheduler has no knowledge of the type of workflow and the number of workflows in advance. Moreover, the workload is constantly changing, so we can view this model as a dynamic process. In this model, each type of workload may have its QoS requirements. Techniques dealing with multiple workflows must have capable of dealing with the dynamic nature of this model to meet the QoS requirements.

TABLE 2.1: Workflow(s) used and evaluation environment

Algorith(s) used	Evaluation Strategy		Montage	CyberShake	Epigenomics	SIPHT	LIGO	Randomly Generated	Other
	Simulation	Real cloud							
Alaei, Mani, et al. [40]	√		√	√	√		√		
Ranjan, Rohit, et al. [41]	√								Google cluster-usage traces
RMFW [42]	√		√	√		√	√		
Li, Chunlin, et al. [43]		√						√	
SPTS [44]		√							Real live video application
OWS-MRL [45]	√			√		√	√		
GL Stavrinides and H D Karatza [46]	√							√	
REEWS [47]	√								GE
Basit Qureshi [48]	√				√				Broadband
CEAS [38]	√		√	√		√	√		

Table 2.1 continued from previous page

Algorith(s) used	Evaluation Strategy		Montage	CyberShake	Epigenomics	SIPHT	LIGO	Randomly Generated	Other
	Simulation	Real cloud							
Monire Safari and Reihaneh Khorsand [49]	√							√	
EDF-DVFS-AC [50]	√							√	
EICB [51]	√							√	
MHRA [52]		√							Embarrassingly Parallel, Matrix multiplication Scatter–Gather
MSMOOA [53]	√		√		√				
EnReal [54]	√								Swinburne astronomy
ERUETB [55]		√						√	
EAS [56]	√							√	
DEWTS [57]	√							√	
ESFS [58]	√		√			√	√		
GTI [59]	√		√				√	√	AIRSN, SDSS
ERAS-D [37]	√							√	

Table 2.1 continued from previous page

Algorith(s) used	Evaluation Strategy		Montage	CyberShake	Epigenomics	SIPHT	LIGO	Randomly Generated	Other
	Simulation	Real cloud							
DVFS-MODPSO [60]	√							√	Neuroscience & protein annotation
EHEFT [61]	√							√	
ECPOP [61]	√							√	
SPSS-EB [62]	√		√			√	√		
SPSS-ED [62]	√		√			√	√		
EES [63]	√							√	GE
Lizhe Wang [64]	√							√	
pSciMapper [65]		√						√	GLFS, Volume Rendering

2.4.3 Algorithm Scheduling Paradigm

There were many comprehensive studies on the taxonomy of task scheduling algorithms in distributed environments. As an example, a classification of scheduling algorithms in general-purpose distributed computing systems introduced in [67], Yu et al. [68] studied the workflow scheduling problem in grid environments, while Kwok and Ahmad [69] established a static scheduling algorithms taxonomy for allocating directed task graphs to multiprocessors, and M A Rodriguez and R Buyya [7] extensively studied scheduling scientific workflow algorithms on Infrastructure-as-a-Service (IaaS) clouds. The scheduling models presented by [7] [66] [67] [68] [69] are most relevant to this survey problem and hence, are identified and considered; summary of surveyed algorithms given in Table 2.3.

This study finds the scheduling models such as task-VM mapping dynamicity, resource provisioning strategy, scheduling objectives, and optimization strategies.

Task-VM mapping is the process of mapping workflow tasks to available VMs. It can be done in two ways, either statically or dynamically. In dynamic mapping, the scheduler maps the tasks to VMs in real-time with constraints at a running time whereas, in static mapping, the scheduler maps the tasks to VMs before task execution.

Resource provisioning is the allocation of cloud resources and services to user-submitted applications. On IaaS clouds, this can consider the pricing model, VM type, and task parallelism. Resource provisioning can be done dynamically or statically. The dynamic resource allocation techniques provision the resources at run-time with considering the performance constraints, whereas static resource allocation techniques calculate and prepare the provision of resources before the execution of the application.

Scheduling objectives Energy optimization is the common objective of the studied algorithms. Further, along with the energy optimization algorithms considered few other optimization objectives such as makespan, workload, monetary cost, reliability, CO_2 emissions, resource utilization, result precision, and service level agreements (SLAs). The scheduling objectives covered in this survey are presented in Table 2.3.

TABLE 2.2: Algorithm classification for the application model

Algorithm(s) used	Workflow Dynamicity		
	Single	Multiple workflows	Ensemble
Alaei, Mani, et al. [40]	√		
Ranjan, Rohit, et al. [41]		√	
RMFW [42]		√	
Li, Chunlin, et al. [43]	√		
SPTS [44]	√		
OWS-MRL [45]	√		
GL Stavrinides and H D Karatza [46]	√		
REEWS [47]	√		
Basit Qureshi [48]		√	
CEAS [38]	√		
Monire Safari and Reihaneh Khorsand [49]	√		
EDF_DVFS_AC [50]	√		
EICB [51]		√	
MHRA [52]	√		
MSMOOA [53]	√		
EnReal [54]		√	
ERUETB [55]		√	
EAS [56]	√		
DEWTS [57]	√		
ESFS [58]	√		
GTI [59]	√		
ERAS-D [37]	√		
DVFS-MODPSO [60]	√		
EHEFT [61]	√		
ECPOP [61]	√		
SPSS-EB [62]			√
SPSS-ED [62]			√
EES [63]	√		
Lizhe Wang [64]	√		
pSciMapper [65]	√		

TABLE 2.3: Scheduling model

Algorithm(s) used	Task-VM mapping dynamicity		Resource Provisioning Strategy		Scheduling objective(s)	Optimization strategy
	Static	Dynamic	Static	Dynamic		
Alaei, Mani, et al. [40]	√		√		energy, makespan, cost, reliability	Metaheuristic
RMFW [42]	√		√		makespan, resource utilization, energy consumption, cost, availability, security	Heuristic
Ranjan, Rohit, et al. [41]	√		√		makespan, execution time, fault-tolerance, energy consumption	Heuristic
Li, Chunlin, et al. [43]	√		√		makespan, energy, resources utilization	Heuristic
SPTS [44]	√		√		makespan, system response time resource utilization, load balancing	Heuristic
OWS-MRL [45]	√		√		makespan, cost, energy, resources utilization	Heuristic
GL Stavrinides and H D Karatza [46]	√		√		energy, SLA, monetary cost, results precision	Heuristic
REEWS [47]	√		√		deadline, energy, reliability	Heuristic
Basit Qureshi [48]	√		√		resource utilization, energy	Heuristic

Table 2.3 continued from previous page

Algorithm(s) used	Task-VM mapping dynamicity		Resource Provisioning Strategy		Scheduling objective(s)	Optimization strategy
	Static	Dynamic	Static	Dynamic		
CEAS [38]	√		√		cost,energy, deadline	Heuristic
Monire Safari and Reihaneh Khorsand [49]	√		√		deadline, SLA, energy	Heuristic
EDF_DVFS_AC [50]	√		√		results precision, SLA, energy	Heuristic
EICB [51]	√		√		deadline, energy	Heuristic
MHRA [52]		√		√	makespan, energy	Heuristic
ERUETB [55]	√		√		resource utilization, energy	Heuristic
MSMOOA [53]		√		√	makespan,cost, energy	Metaheuristic
EnReal [54]	√		√		resource utilization, energy	Heuristic
EAS [56]	√		√		Deadline and Energy	Heuristic
DEWTS [57]	√		√		deadline, energy	Heuristic
ESFS [58]	√		√		Deadline and Energy	Heuristic
GTI [59]	√		√		deadline, energy	Heuristic
ERAS-D [37]	√		√		deadline, cost, energy, CO_2 emmisions	Heuristic
DVFS-MODPSO [60]	√		√		makespan, cost, energy	Metaheuristic
EHEFT [61]	√		√		makespan, energy	Heuristic

Table 2.3 continued from previous page

Algorithm(s) used	Task-VM mapping dynamicity		Resource Provisioning Strategy		Scheduling objective(s)	Optimization strategy
	Static	Dynamic	Static	Dynamic		
ECPOP [61]	√		√		makespan, energy	Heuristic
SPSS-EB [62]	√		√		deadline, workload, energy	Heuristic
SPSS-ED [62]	√		√		deadline, workload, energy	Heuristic
EES [63]	√		√		deadline, cost, energy	Heuristic
Lizhe Wang [64]	√		√		makespan, energy	Heuristic
pSciMapper [65]		√		√	makespan, energy	Heuristic

i) Energy minimization: This chapter aims to study the energy-aware workflow scheduling techniques in a cloud environment. The public and private organizations globally have been developing increased attention to reduce the ecological footprint. This issue has not attracted the cloud field to create various techniques to minimize the energy consumption of cloud data centers. As we discussed in Section 2.1, the need for energy conservation in this field is attracting research.

ii) Monetary Cost: While scheduling workflows in a cloud environment, the financial cost of the workflow executions is the price that users need to pay to the cloud service providers (CSPs) because of using cloud resources and is concerned by both the customers and CSPs [70]. This represents the restriction on the overall cost of executing all tasks. The algorithms we considered in this survey balance the cost while considering some performance or non-functional concerns objectives such as energy consumption and reliability.

iii) Makespan: The makespan of the workflow is stated as the length of the schedule from the start of the workflow entry task(s) to finish the exit task. As with the cost, it is concerned by both the customers and CSPs [70]. In most of the surveyed algorithms, it minimized by defining some time limit set by the user or user-defined deadline.

iv) Reliability-Aware: Reliability is the capability of a system or component to perform its defined functions according to the declared conditions for a prescribed period. The overconsumption of energy in data centers leads to system reliability issues while increasing operational costs and CO_2 emissions. The aim of some of the algorithms in this survey is to maintain dependency constraints, minimizing energy consumption, and maximizing the reliability of the workflow execution while meeting the user-specified QoS constraint.

v) CO_2 Emissions-Aware: Due to the increasing deployment of cloud data centers across the world, consuming massive amounts of electrical energy makes data centers unsustainable. As stated in Section 2,1, it is essential to reduce CO_2 emissions for data centers to make them environment-friendly.

vi) Maximize resource utilization: The cost-aware, energy-aware, and workload optimization algorithms indirectly address this objective by scheduling the idle gaps of leased VMs. The empty time slots on leased VMs not uncommon for workflows applications due to dependencies and performance requirements.

vii) Results precision: Real-time workflow applications have strict timing and performance needs. The algorithms which aim to provide timeliness and energy efficiency divide each task into the mandatory part that produces near the result of the minimum tolerable precision and optional part to enhance the results of the mandatory part [71].

viii) Workload optimization: The scheduling techniques that are capable of dealing with workflow ensembles strive to maximize the number of workflows executed, which is a constraint with user-defined deadlines and budgets.

ix) Service-Level-Agreements (SLAs): Cloud services are interrelated with SLAs. In general, SLAs include a set of services the provider will deliver to customers. These include a comprehensive definition of each service, accountability of the provider to the consumer, a metric system to measure each service whether the provider is offering the services as ensured, and a service auditing mechanism. Remedial measures are available to the consumer and the provider if the contractual terms are not satisfied, and state how the SLA will change over time.

Optimization Strategy Due to ample solution space, it takes a long time for finding the best possible solution, hence scheduling problems in clouds is NP-hard and even it is expensive for optimal solution small-scale problems [7]. Casavant and Kuhl [67] identify four optimization techniques for task scheduling in distributed environments a) solution space enumeration and search b) graph theoretic c) mathematical programming and d) queuing theoretic. Mathematical programming, queuing theoretic, and solution space enumeration and search are the most appropriate model for our problem. We identified three scheduling strategies in the surveyed algorithms such as heuristic, meta-heuristic, and hybrid algorithms.

Heuristic approach: When classic approaches are too time-consuming to find a solution to a scheduling problem, then the heuristic approach is a good solution. It is simply a set of rules to find an optimal solution to an individual problem [72]. The rules are designed in such a way that they very particular to the problem and can produce an acceptable solution. The heuristics for the scheduling considered in this work use the characteristics of cloud resources and workflow problems to discover a scheduling technique to minimize energy consumption by meeting the QoS requirements. The benefits of the heuristic approach are

easy to implement, performs, produces a near-optimal solution within a time limit, and predictable in nature.

Meta-Heuristic approach: Unlike heuristics, meta-heuristic approaches are general-purpose techniques developed to solve optimization problems [72]. These higher-level policies are used for finding near-optimal results by implementing a problem-specific heuristic . Unlike heuristics, these are computationally intensive and take more time to run the application but produce desirable solutions. In particular, for workflow problems, meta-heuristic-based algorithms ensure near-optimal solutions [73].

Hybrid algorithms: Originally, researchers focused on developing heuristic scheduling strategies to deal with the workflow applications executed in distributed environments. However, due to the limitations of heuristics in getting near-optimal solutions and the uncertain amount of time required, researchers focused on meta-heuristics. Further, to meet the workflow scheduling challenges, hybrid approaches were introduced. Hybrid approaches are better in performing many-objective workflow scheduling problems in clouds due to their convergence speed and accuracy. Meta-heuristic and heuristic strategies perform better to solve the workflow scheduling problems. By one of the combinations, the solution accuracy will be improved: i) two meta-heuristics and ii) a heuristic with some pre-defined rules and meta-heuristics.

Originally, the researchers are fascinated in developing the heuristic scheduling strategies to deal with the workflow applications to execute in distributed environments. But, due to limitations of heuristics in getting a near-optimal solution and uncertain time getting such solution researchers focused on meta-heuristics. Further, to meet the workflow scheduling challenges hybrid approaches were introduced. The hybrid approaches are better in performing many-objective workflow scheduling problems in clouds due to their convergence speed and accuracy. Meta-heuristic and heuristic strategies perform better to solve the workflow scheduling problems. By one of the combinations, the solution accuracy will be improved: i)two meta-heuristics and ii) a heuristic with some pre-defined rules and meta-heuristic.

2.4.4 Resource model

In this section, we discussed the leased VMs (resource) model considerations and assumptions of surveyed algorithms and summarized them in Table 2.4. We included different concerns of resource (VMs) model such as leasing, type, pricing, delays, and core count. The algorithms in this work not considered VM delays and assumed a static pricing model.

VM Leasing model The algorithms assumed that the CSPs provide unlimited VMs or they have a cap (limited) on the number a user is allowed to lease. With an unlimited VM provisioning method, applications should have an efficient scheduling mechanism to manage this plethora of resources. Whereas with limited, resource provisioning problem is simple.

VM Type equality As discussed in Section 3.1, the CSPs provide various VM instances. To simplify the scheduling process algorithms consider a single type VM, which unsuccessful in getting benefited from the heterogeneous nature of cloud resources, while algorithms with multiple VM typeleasing strategies produce efficient results.

VM Pricing model In the surveyed algorithms, a static VM pricing model is considered. The pricing of the VMs varies from provider to provider. General pricing schemes we found such as dynamic is the real-time pricing model, static the service provider sets the price and remains constant, time unit VMs are charged per unit time, such as Amazon charges per hour, subscription-based price based on the subscription period and hybrid pricing where price changes according to the job queue wait times (static/dynamic) [74].

VM Delays The VM provisioning delays are estimated by the algorithms to make accurate scheduling. This process is very much essential in the case of real-time workflow applications where applications demand good response times. Many factors affect this delay in a cloud environment, such as the virtualization layer due to a large number of machines running on the same physical machine (PM). These delays are non-negligible and highly inconsistent. The VM provisioning policy affects the scheduling objectives such

TABLE 2.4: Techniques classification based on resource model

Algorithm(s) used	VM Leasing model		VM type equality		VM core count	
	Limited	Unlimited	Single	Multiple	Single	Multiple
Alaei, Mani, et al. [40]	√			√		√
Ranjan, Rohit, et al. [41]	√			√		√
RMFW [42]	√			√	√	
Li, Chunlin, et al. [43]	√		√		√	
SPTS [44]	√		√		√	
OWS-MRL [45]	√			√		√
GL Stavrinides and H D Karatza [46]	√			√		√
REEWS [47]	√			√		√
Basit Qureshi [48]	√			√	√	
CEAS [38]		√		√	√	
Monire Safari and Reihaneh Khorsand [49]		√		√	√	
EDF_DVFS_AC [50]	√			√		√
EICB [51]		√		√	√	
MSMOOA [53]		√		√	√	
EnReal [54]		√	√		√	
MHRA [52]		√	√			√
ERUETB [55]		√	√		√	
EAS [56]		√		√	√	
DEWTS [57]		√	√		√	
ESFS [58]		√		√	√	
GTI [59]		√	√		√	
ERAS-D [37]		√	√		√	
DVFS-MODPSO [60]		√		√	√	
EHEFT [61]		√	√		√	
ECPOP [61]		√	√		√	
SPSS-EB [62]		√	√		√	
SPSS-ED [62]		√	√		√	
EES [63]		√	√		√	
Lizhe Wang [64]		√	√		√	
pSciMapper [65]		√	√		√	

as cost and makespan of the workflow applications [75, 76]. The VM de-provision delays affect the execution costs of workflows applications if not shut down the VM within the billing period (VM time unit).

VM Core-count This classification refers to algorithms assumed to get provisioned single-core VMs or multi-core VM to schedule single tasks or multiple tasks simultaneously on them. With single-core machine algorithms capable of scheduling one task at a time which makes the scheduling policy simple. Whereas multi-core VMs algorithms schedule multiple tasks simultaneously, which possibly reduces cost, energy, and free from intermediate data transfer. At the same time, this simultaneous scheduling of tasks competes for the shared resources which results in performance degradation.

2.4.5 Energy optimization techniques

In this section we discussed various energy optimization methods which we identified in the survey such as Sequences tasks merging, Parallel tasks merging, VM reuse, Task slacking, DVFS, Per-core DVFS, Task Migration, Clustering of Tasks/ Workloads, VM power utility, VM Placement, Limiting CPU utilization, PM mode switch, and Pareto optimality. The summary of the energy optimizations techniques is tabulated in Table 2.5.

2.4.5.1 Sequences tasks merging

In a workflow, any two sequence tasks such as t_i and t_{i+1} where t_i is the only predecessor of t_{i+1} and t_{i+1} is only successor of ti can be mapped to different VMs of vm_i and vm_{i+1} respectively. We can merge tasks t_i and t_{i+1} into single task to map onto an optimal VM of type vm_k. This process have threefold benefits: 1) the merged tasks won't affect the tasks' precedence constraints; 2) significant reduction in execution times of tasks; 3) We still merge the tasks even if execution cost of t_i and t_{i+1} on vm_k is equal to the sum of execution times of t_i on vm_i and t_{i+1} on vm_{i+1} as we do not need to transfer big dataset in some cases, which can save a large amount of valuable energy along with the communication time. The process of sequence task merging is shown in Figure 2.2.

FIGURE 2.2: The process of sequence task merging

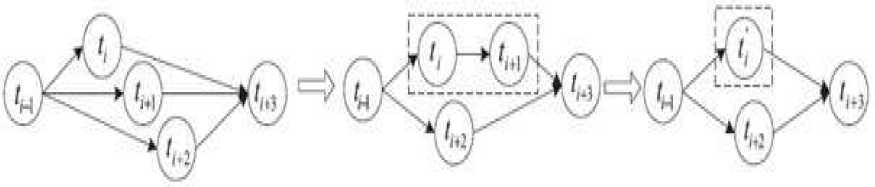

FIGURE 2.3: The process of parallel task merging

2.4.5.2 Parallel tasks merging

Workflow as it may be a complex structure for this reason workflow scheduling in distributed environment is a difficult problem. We regard any two or more tasks with the same predecessors and successors as the parallel tasks. Usually, the parallel tasks start their execution at the same time but might finish at different times. As an example, consider three parallel tasks t_i, t_{i+1} and t_{i+2} with execution times 30, 20 and 60 minutes respectively and the predecessor task t_{i-1} and successor task t_{i+3}. As t_i, t_{i+1}, and t_{i+2} are parallel tasks hence task t_{i+3} cannot start its execution until the longest task t_{i+2} completes its execution. If we execute t_i and t_{i+2} sequentially on optimal VM then the execution time is 50 minutes which is less than the execution time of t_{i+2}, and separate VM is used for t_{i+2}, the whole process will not delay the start time of t_{i+3}. If we merge and execute tasks t_i and t_{i+1} as in *sequences tasks merging*, then we can achieve the same benefits as in Sequences tasks merging [38]. The process of parallel task merging depicted in Figure 2.3.

2.4.5.3 VM Reuse

The cloud providers such as Amazon offer different VM instances to the customers such as small, medium, large, and x.large. These instances vary in computing and storage capacities hence available at a different price per unit of time. For simplicity, consider VM lease unit time is one hour, i.e., if a VM used more than 60 minutes and less than 120 minutes, then it will be charged for two hours. If tasks are computed on a VM that takes 70 minutes, then service will be charged for two hours. Here, the remaining gap is 50 minutes of unused VM that still must be paid to the provider. Active but in an idle state the processors consume up to 70% of the power of its peak load [77]. Most heuristics can try to utilize this idle gap to schedule the new tasks which are fit to the idle gap [46, 38, 49, 57, 61] and in some cases leased time will be extended to schedule long tasks to minimize the computing cost and as well as energy consumption [38].

2.4.5.4 Task slacking

In the context of workflow scheduling, non-critical tasks have slack time (the idle time slots of leased VMs). During the slack time, VMs consume a significant amount of power. Reclaiming the slack time by dynamically reducing the operating voltage and frequency is the most used technique in workflow scheduling to lower the unnecessary energy consumption of leased VMs.

2.4.5.5 DVFS

Dynamic Voltage and Frequency Scaling (DVFS) is a widely-known approach that has commonly been used to make the processor energy-efficient [78]. The DVFS technique allows the processors to scale the supply voltage and as well operating frequency dynamically, based on the workload conditions. Most of the modern processors are DVFS enabled.

2.4.5.6 Per-core DVFS

It is similar to DVFS, but here the concept is applied to multi-core systems. Modern processors with multi-core architectures have integrated voltage regulators for each of its core, enabling per-core DVFS [79]. That is, each core of the processor can operate at a different voltage and frequency level than the other cores of the same processor. While this provides flexibility and better energy efficiency, it involves great control complexity, particularly in the case of clouds where the heterogeneous physical resources are virtualized and managed by the hypervisor (Virtual Machine Monitor – VMM) such as Xen, Hyper-V, and VMware ESXi.

2.4.5.7 Task Migration

Task migration enables effective resource utilization, improved performance through load balancing, and energy efficiency. Inefficient processors are identified by evaluating the performance metric called the ratio of effectiveness (RE) of the assigning task on the processor. RE can define as the ratio of the active time slot, which runs a specific task on the selected processor to the scheduled length of all the tasks (makespan). Processors with low RE are wasting energy with idle slots are shutdown and tasks assigned to it are migrated to better RE processors which are power efficient.

2.4.5.8 Clustering of Tasks/ Workloads

A complex workflow application contains many tasks which are mapped to different processors or VMs to perform their job. These tasks need to communicate with each other by transferring a huge amount of data, which consumes a lot of energy and as well as incurs communication costs. Clustering of tasks or workloads is the technique that reduces these inter-process communication energy overheads [47]. Clustering is performed to group the tasks (workloads) which execute a similar job, and these tasks need to communicate with each other while executing, and hence clustering saves the loads of transmitting the data.

2.4.5.9 VM Power utility

The power utility concept of cloud resources is considered by some researchers to address the energy consumption optimization problem of cloud workflow scheduling applications. The power utility of computing resources (VMs) can be defined as the workload finished per unit of energy. The power utility of any task t_i on VM of type k is the ratio of the workload (w_i) of t_i to the total energy consumed by the task t_i to execute on vm_k. The task consumes less power on the selected *vm* if the power utility factor is more; we can call such *vm* as optimal. The scheduling heuristic used to map the tasks to the resources can select the optimal *vm* if it meets QoS requirements.

2.4.5.10 VM Placement

The process of selecting the most suitable Physical Machine (PM) for the VM is called VM Placement. The mapping of VMs on PM allowed changing according to the workload, or it fixed throughout the process. The first one, called dynamic VM placement, and the latter is static VM placement. The successful placement of VMs on PMs can improve the performance of the cloud, resource utilization, and reduced energy consumption. The effective VM placement heuristics maximize the active PMs which save energy [48].

2.4.5.11 Limiting CPU utilization

The energy consumption for PMs is composed of CPU, memory, disk, power supply, etc. Various research results in the literature show that CPU utilization significantly affects energy consumption [80]. The relationship between CPU utilization and energy consumption is non-linear. Energy consumption of CPU at full workload is more than twice 50% of CPU workload. We can get benefited from energy saving if we limit the CPU utilization under some threshold.

2.4.5.12 PM mode switch

Underutilized Physical Machines of data center consumes a significant amount of power. Each PM can be operated in one of the three different modes such as active mode, low-power mode and sleep mode. By switching the underloaded PMs to the low-power mode or sleep mode, we can minimize energy. Depending on the hosted application service time, Xu et al. [54] propose two power modes for the PMs with all idle VMs i.e., low-power and sleep modes.

2.4.5.13 Pareto optimality

Getting feasible solutions to the multi-objective problem is a challenging task. One of the ways to find such solutions is the Pareto efficiency. The "Pareto solutions" states that optimization in one objective might occurs with the worsening of solutions of at least one of the remaining individuals. Therefore, rather than a unique solution, a set of Pareto points is preferable for multi-objective problems [81].

TABLE 2.5: Techniques used to optimize energy

Algorithm(s) used	Tasks merging		DVFS	Task			VM			Limiting CPU utilization	PM mode switch	Pareto optimality	Others
	Sequence	Parallel		Slacking	Migration	Clustering	Reuse	Power utility	Migration				
Alaei, Mani, et al. [40]								√					Adaptive fault detector design based on improved (IDE) algorithm.
Ranjan, Rohit, et al. [41]													Containerized virtualization.
RMFW [42]			√						√	√			
Li, Chunlin, et al. [43]										√			
SPTS [44]													Shortest path algorithm.
OWS-MRL [45]													Resource provisioning using reinforcement learning.
GL Stavrinides and H D Karatza [46]				√			√						
REEWS [47]			√			√							
Basit Qureshi [48]						√			√				
CEAS [38]	√	√	√	√			√						
Monire Safari and Reihaneh Khorsand [49]			√	√			√						
EDF_DVFS_AC [50]			√	√									
EICB [51]					√	√				√			
MHRA [52]								√				√	
MSMOOA [53]												√	
EnReal [54]											√		
ERUETB [55]			√		√								
EAS [56]	√							√					
DEWTS [57]				√			√						
ESFS [58]			√	√									
GTI [59]			√										
ERAS-D [37]			√										
DVFS-MODPSO [60]			√									√	
EHEFT [61]					√		√						
ECPOP [61]					√		√						
SPSS-EB [62]								√					
SPSS-ED [62]								√					
EES [63]				√									
Lizhe Wang [64]			√	√									
pSciMapper [65]						√							

2.4.6 Summary of surveyed algorithms and future directions

In this section, we evaluated and reviewed the algorithms, to highlight the key idea, advantages, and trade-offs of each technique. We suggested future research directions for each algorithm. As discussed in Section 2.3, the common main objective of the algorithms is to reduce energy consumption for scheduled workflow without compromising the performance and considering other objectives such as cost, deadline, reliability, etc. The methodology used for an efficient schedule of tasks and resources to meet certain objectives is summarised under the key idea of the algorithm. The possible further enhancement of algorithms to make it more efficient along with the key idea, advantages, and disadvantages summarized in Table 2.6.

TABLE 2.6: Comparison of Energy-Aware Techniques

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
Alaei, Mani, et al. [40]	2021	A multi-objective fault-tolerant framework: 1. Monitor to collect tasks and resource load 2. Future workload analyzer 3. A fuzzy planner to map tasks to efficient resources with proactive and reactive fault-tolerance 4. Planner used IDE algorithm	1. A higher degree of fault tolerance 2. Optimal makespan, energy consumption, and task fault ratio 3. Reduces overall cost.	This work considered only the VM faults. However, the reliability issues can be caused by many other factors like network and I/O.	More energy can be saved by switching off underloaded and idle hosts
Ranjan, Rohit, et al. [41]	2020	1. A CaaS model is proposed for energy-efficient workflow scheduling 2. Containers are created and used to execute different tasks	1. Containers require fewer system resources because they don't include operating system images 2. Low migration overheads 3. Great increase in application scalability	1. Placing dependencies on containers that can limit portability between servers 2. Incurs an expensive performance overhead	System performance can be improved by introducing prediction method and performance modeling for microservice based applications

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
RMFW [42]	2020	1. Multiple reinforcement learning (RL) agents proposed on a framework that cooperates for better task scheduling and resources utilization 2. DVFS used for powersaving	1. Cooperative RL-based agents improve the execution time by avoiding a greedy approach in scheduling 2. Load-balancing 3. Lower costs and energy consumption by increasing resource utilization.	The lower threshold used for resource utilization is very low	Energy-saving can be improved by using evolutionary algorithms for VM consolidation.
Li, Chunlin, et al. [43]	2020	1. Queuing model-based job scheduling for better response times 2. Load-balancing for geographically distributed clouds 3. Shortest path algorithm for energy saving	1. M / M / C queue model gives better response and reduces resources wastage 2. Significant energysaving	1. Job arrival queue rate is not predictable. It may discourage other jobs from entering into it 2. The shortest path algorithm always not find the best path	By considering diverse VM types might observe increased performance in energy consuption

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
SPTS [44]	2020	1. Workflow task partitioning algorithm used based on hypergraph partitioning 2. Task scheduling problem is modeled as the shortest path problem (SPP) 3. SPP solved by using Dijkstra algorithm	Significant saving in completion time and energy consumption	The shortest path algorithm always not find the best path	Cost and energy consumption can be further reduced by reusing the virtual machines in the leased clusters
OWS-MRL [45]	2020	1. Clustering of resources for mapping to appropriate workflows 2. Reinforcement learning method used to allocate resources to tasks 3. Limiting the use of resources	1. Multi-agent environment achieved a global optimal solution 2. Significant saving in cost and power utilization	Operating the resources at minimum frequency induce transient errors	An agent which handles the resources allocation to workflow should maintain a trade-off policy between power utilization and quality of a schedule

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
G L Stavrinides [46]	2019	1. Tasks prioritized based on the Earliest Deadline First (EDF) 2. Priority ties beaked using the highest average computing cost 3. Tasks mapps to per-core DVFS enabled VMs	1. Energy efficient because of per-core DVFS VMs 2. Assured QoS using EDF -based Task priority 3. Cost-effective utilizing idle time of VMs	1. If a task has many parents with input error, then it may produce unacceptable results 2. Tasks may not fit in schedule gaps of VMs	Result precision can improve by considering the tardiness tolerance situations.
REEWS [47]	2019	1. Task prioritization 2. Clustering tasks 3. Deadline distribution 4. Applied DVFS to reduce energy consumption	1. Communication cost minimized 2. Crucial tasks given highest priority in scheduling 3. Rescues the tasks starving from the higher priority	More than one scheduling order possible because of topological order of tasks	1. Make more energy efficient using per-core DVFS 2. Improved system reliability through load-balancing

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
Basit Qureshi [48]	2019	1. Task ranks calculated based on earliest finish time and lowest price of jobs in a queue 2. Tasks are mapped to VMs in priority order	1 Efficient VM placement 2. Load balancing 3. Maximization of idle VMs for energy efficiency 4. Polynomial time	Limited to CPU intensive workloads	Better cost estimation includingthe latency of network, memory and storage
CEAS [38]	2018	1. Optimal VM selection 2. Task merging 3. VM reuse 4. Task slacking	1. Improved makespan. 2. Reduced execution cost. 3. Energy-efficient 4. Polynomial time 5. Suitable for a commercial multi-cloud environment.	1. In VM reuse still chances for an unused time slot 2. Extensive coding	1. Cost saving considering VM delays 2. Enenrgy saving by effectively utlizating gap between makespan and deadline
Monire Safari and Reihaneh Khorsand [49]	2018	1. Sub-deadline distribution to sub-workflows 2. Ordered tasks based on distributed deadlines are mapped to VMs 3. A proper frequency is set for the selected resources	1. Guaranteed optimal execution time. 2. Assured SLAs irrespective of a number of tasks 3. Polynomial-time heuristics	1. Insignificant energy saving with less number of processors 2. Inefficient resource utilization with less number of processors.	We can minimize energy by optimizing data communicating processors

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
EDF_DVFS _ AC [50]	2018	1. Tasks prioritized based on the EDF 2. Priority ties beak using the highest average computational cost 3. Optimal VMs selected for execution 4. If schedule gaps then apply DVFS to save energy	1. To fill schedule gaps used per-core DVFS on the underlying heterogeneous environment 2. Improved SLA violations 3. Improved energy saving	A slight loss in result precision of computationally intensive tasks	Better energy saving with live VM migration and improve system reliability
EICB [51]	2018	1. Grouping instances for batch processing 2. Resource monitoring by calculating the resource usage ratio VM to PM 3. Resource allocation in energy-efficient way	1. More energy reduction with increased task instances 2. Increased response with load-balance 3. Reliable for real-world applications	If the CPU threshold is low then the loss of energy by opening large PMs. Care should take to set threshold.	To save more energy a) Applying DVFS on newly opened idle resources b) Considering VM core-level threshold

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
MHRA [52]	2018	Minimizes energy and execution time by considering: 1. Sequence setup times between tasks 2. VM setup and down time 3. System architecture energy profile 4. Resource energy performance factor	1. Polynomial time Multi-heuristic resource allocation algorithm 2. A subgroup of dependency free tasks executed in parallel	1. Drastic makespan increase If importance factor >+.7 2.Execution environment not open for research 3. VM idle times not considered	More energy saving using DVFS for VM idle slots
MSMOOA [53]	2017	1. A manager adopted to manage the resources of the data center 2. Multi-swarm is used to find personal best, global best and swarm best	1. The adopted resource manager avoids the influence of unstable resources 2. Efficient in searching non-dominated solutions in a cloud-environment	Not using the scalable nature of cloud and works with stable resources	1. Workload migration to achieve stable resources 2. Select optimal swarm by making the manager server dynamic

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
EnReal [54]	2016	Live VM migration from the under-loaded PMs and PM mode switching	1. Resource utilization better than existing works 2. Excellent performance with increased workflow dynamicity	Live migrations increase memory cost	Memory optimization can save energy and cost
ERUETB [55]	2016	1. Cloud module consolidation by setting the threshold to VMs and allowing concurrent execution 2. Workload migration to achieve stability 3. VM reuse	1. Avoids placement of new VMs by concurrent execution of modules 2. Efficient resource usage 3. Inprove energy saving	Forward procedure mapping schemes suffer from poor resource utilization ratio compared to backward scheduling	Minimizes unnecessary workload migrations by making VM threshold cap dynamic
EAS [56]	2016	1. Tasks-VMs map based on power utility factor by considering the deadline 2. Merging sequential tasks with common successor 3. Map such task to a VM	Sequential task merging and mapping to common VM will reduce energy and communication cost	1. Tasks with less power utility the factor will fail to meet deadlines 2. VM idle times	Better power utility factor estimation by distributing sub-deadline to tasks

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
DEWTS [57]	2014	1. Prioritizes tasks and makespan using HEFT 2. Merging inefficient VMs 3. Slack estimation to apply DVFS	1. VMs optimization reduces energy consumption 2. Leased time slot reuse	Smaller DAG length with higher parallelism but in reality workloads unpredictable	Parallel task merging can save energy and cost
ESFS [58]	2014	1. Initial assignment with HEFT 2. Scales the assigned frequencies 3. Slack distribution in an energy-efficient way	1. Proper resource utilization irrespective of the number of hosts 2. The balance between execution time and energy consumption	1. Comparatively, makespan is more with less number of hosts (<16) on LIGO 2. Task's frequency scaling does not necessarily bring energy efficiency	1. Job sensitivity consideration towards frequency scaling 2. Considering networks characteristics to minimize communication cost
GTI [59]	2014	1. For idle hosts scale the frequency to the lowest level 2. Measuring energy saving ratio 3. Used HEFT makespan while operating at minimum frequency as deadline base	1. Energy-efficient ignoring settings of various hosts and deadline constraints 2. Communication to computation ratio (CCR) effectively works to reduce cost 3. Assured QoS	The energy reduction not significant with deadline ratio 0.7and 0.8 on SDSS workflow	Performance evaluation on real DVFS-enabled processors

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
ERAS-D [37]	2013	Optimal resource utilization through backward task scheduling and DVFS	1. Decreased energy cost, energy consumption, and CO2 emission 2. Improves provider profit	A separate algorithm developed for compare results	1. Workload migration to increase system reliability 2. CPU mode switching to improve energy saving
DVFS-MODPSO [60]	2013	Provides non-dominated (Pareto) solution through 1. Voltage and frequencies (swarm) initialized with random values 2. HEFT applied to minimize makespan 3. For each particle new parameters like velocity and position calculated then updated tili termination	Averts the compromise between performance and energy consumption by handling various QoS requirements.	Significant improvement in cost and energy consumption but not in makespan	Deadline distribution to sub-workflows to achieve improvement in makespan for synthetic workloads

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
EHEFT [61]	2013	Inefficient processor shutdown by calculating the performance metric RE (Ratio of Effectiveness)	1. Reduce time and energy in parallel applications 2. Works well for communication and computation-intensive applications	Load balancing issues. Some resources overloaded while some idle	1. Workload load balancing 2. Can make more energyefficient using DVFS processors
ECPOP [61]	2013	Inefficient non-critical VM shutdown by calculating the performance metric RE (Ratio of Effectiveness)	1. Reduce time and energy in parallel applications 2. Works well for communication and computation-intensive applications	Slack peroids of VMs not effectively used	Can make more energy efficient using DVFS processors
SPSS-EB [62]	2013	Static Provisioning-Static Scheduling (SPSS) model ensured energy and deadline constraints using SPSS-EB and SPSS-ED algorithms respectively	1. A small number of VM selection 2. Efficient resource utilization for energy constraints 3. Resources provision for the ensemble allows better resource utilization	1. VM migration not supported 2. VM limitations 3. Not support for heterogeneous computing	1. Communication cost consideration 2. Considering heterogeneous nature of cloud resources

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
SPSS-EB [62]	2013	Static Provisioning-Static Scheduling (SPSS) model ensured energy and deadline constraints using SPSS-EB and SPSS-ED algorithms respectively	1. A small number of VM selection 2. Efficient resource utilization for energy constraints 3. Resources provision for the ensemble allows better resource utilization	1. VM migration not supported 2. VM limitations 3. Not support for heterogeneous computing	1. Communication cost consideration 2. Considering heterogeneous nature of cloud resources
EES [63]	2012	1. Slack room study 2. Achieve global optimality by schedule slack tasks on nearby VM which running on uniform frequency	1. Reduced energy of data centers running parallel applications 2. Assured QoS 3. Used real heterogeneous processors	The original methods selected for results, a comparison is not suitable for heterogeneous computing.	1. VM reuse to minimize cost and energy 2. Communication cost reduction through task merging

Table 2.6 continued from previous page

Algorithm	Year	Key Idea	Advantages	Limitations	Future Research Direction
Lizhe Wang [64]	2010	Reduces the energy consumptio by extending the execution time for the non -critical tasks based on the analysis of the slack time without extending the overall execution time	Green SLA-based approach	1. Increased task execution time 2. Little bit increase in communication cost 3. Chances of idle time of VMs	1. Task merging to minimize communication cost 2. VM reuse to reduce cost
pSciMapper [65]	2010	1. Study of power consumption by resource usage 2. Correlation investigation of workloads, power, and performance (dimensionality reduction)	1. >50% power saving with nominal performance down 2. pSciMapper practical even large-scale workflows 3. Scheduling overheads are less	1. 10-15% performance slowdown not acceptable in some applications 2. Real power values not measured	Performance increasing by mapping tasks to optimal VMs

2.5 Conclusion

Cloud computing is the key technology in digital transformation. Nearly every online end-users use it directly or implicitly. Nevertheless, the customers and cloud service providers face challenges such as energy consumption, monetary cost, QoS, security, load balancing, and SLA compliances. Energy efficiency is one major challenge due to the enormous power consumption of cloud data centers and the emission of greenhouse gases into the environment. This chapter studies various state-of-the-art techniques developed for workflow task scheduling in cloud computing environments to minimize energy consumption by meeting certain objectives. More particularly, its emphasis on techniques considering workflow applications modeled as "DAGs" and where resources are modeled as in the public clouds. The existing works in "energy-aware workflow scheduling" are studied and classified in a way that a reader can get more insight. We believe that this survey will help to shape the future research direction in the field of energy-efficient workflow scheduling in cloud computing.

Several factors such as storage cost, resource latency, network characteristics, and energy consumption by other ITC equipment such as the disk, memory, and thermal parameters needed to consider for a better estimation of the cost and energy consumption.