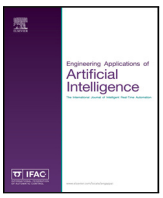
Engineering Applications of Artificial Intelligence 116 (2022) 105345

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai



Survey paper

Resource scheduling methods for cloud computing environment: The role of meta-heuristics and artificial intelligence

Rajni Aron [a,*], Ajith Abraham [b,c]

[a] MPSTME, NMIMS University, Mumbai, India
[b] Machine Intelligence Research Laboratories, Auburn, WA 98071, USA
[c] Center for Artificial Intelligence, Innopolis University, 420500 Innopolis, Russia

ARTICLE INFO

Keywords:
Cloud computing
Resource provisioning
Scheduling
Heuristic methods
Scheduling algorithms

ABSTRACT

The growth and development of scientific applications have demanded the creation of efficient resource management systems. Resource provisioning and scheduling are two core components of cloud resource management systems. Cloud resource scheduling is the most critical problem to solve efficiently due to the heterogeneity of resources, their inter-dependencies, and unpredictability of load in the cloud environment. In this paper, we review the background of scheduling and state-of-the-art scheduling techniques in cloud computing. We first introduce the general background, and phases of scheduling. A comprehensive survey of existing resource scheduling problems proposed so far is presented considering high-level taxonomy. This high-level taxonomy considers Virtual Machine (VM) placement, Quality of Service (QoS) parameters, heuristic methods, and other miscellaneous techniques for resource scheduling. This study also discusses scheduling in Infrastructure as a Service (IaaS) clouds and comparison based on important parameters is also investigated. The importance of meta-heuristic methods and artificial intelligence for resource scheduling methods in cloud computing is discussed thoroughly. The objective of this work is to help the researchers to understand the basic concepts related to scheduling and facilitate the process of designing new scheduling methods by addressing issues raised in the scheduling and studying the existing methodologies.

1. Introduction

Cloud resource management systems provide the facility of resource provisioning as per the demand of scientific application's execution. Foster et al. (2008) defined cloud computing as it provides the facility of the cloud users to use the resources over the Internet, for the execution of their applications as per the requirement. The management of resources in cloud environment is an NP-hard problem. Thus, it is challenging (Xhafa and Abraham, 2010). A low-quality resource management system can potentially result in a long processing time and high cost. The main goal of the cloud resource management systems is to interact with the underline hardware infrastructure and control the provisioning and scheduling of the resources for the successful execution of the application as per the user's requirements. Resource scheduling has become an important core field of cloud resource management system as it is responsible for orchestrating the resources to both cloud providers and cloud users. Resource scheduling algorithm plays an important role in the process of managing the huge number of resources of cloud data centers in an efficient manner to achieve the target of quality requirements with the help of resource monitoring, by managing the decision of resource-job mapping process. Cloud service provider Amazon allows cloud users to lease different types of instances by using elastic compute cloud and storage volumes by elastic block storage and storage object with the help of simple storage service on demand. Resource prices are varied as per the different pricing models (Irwin et al., 2010). In a shared environment, users may compete for resources to meet their timing deadlines and budget too. In such a scenario, resource scheduling plays a critical part in meeting an application's expectations (Shamsi et al., 2013).

There is a need to consider key points such as fair resource allocation and execution of application at the time of problem formulation of cloud resource scheduling (Lee and Katz, 2011). In this paper, basic building blocks of scheduling are presented. It describes the basic characteristics of scheduling and phases of scheduling in cloud computing. Scheduling in different Infrastructure as a Service (IaaS) clouds is presented. Then, we focus on heuristic, meta-heuristic, and hyper-heuristic methods. Heuristic and meta-heuristic methods are apparent for solving many computationally hard problems in an efficient manner (Xhafa and Abraham, 2010). The main intent of this study is to show the usefulness of heuristic methods, especially for scheduling and resource allocation problems in the cloud computing domain. The

* Corresponding author.
E-mail addresses: rajni3387@gmail.com, rajni@nmims.edu (R. Aron).

https://doi.org/10.1016/j.engappai.2022.105345
Received 4 May 2022; Received in revised form 5 August 2022; Accepted 15 August 2022
Available online xxxx
0952-1976/

importance of meta-heuristic methods is discussed in detail. An analysis of heuristic methods is done to show the out-performance of these methods in comparison to the traditional scheduling techniques. The survey of scheduling techniques done in research articles (Tsai and Rodrigues, 2014; Zhan et al., 2015; Arunarani et al., 2019) is mostly based on schemes.

Also, the study conducted by Lucas-Simarro et al. (2013) Zhan et al. (2015) Kumar et al. (2019) Houssein et al. (2021), Konjaang and Xu (2021), Singh et al. (2022) considers only meta-heuristic methods while presenting the classification of scheduling techniques. In Lucas-Simarro et al. (2013), the authors have discussed meta-heuristic algorithms for solving the scheduling problems by placing them in a unified framework. The two major factors, i.e., intensification and diversification are considered in the survey to study about meta-heuristic in cloud computing. In Zhan et al. (2015), scheduling in the application layer, virtualization layer, and deployment layer are focused. They have emphasized on layer wise scheduling method using evolutionary techniques. Resource scheduling on the basis of scheme and types are not considered. In Arunarani et al. (2019), task scheduling related algorithm and performance matrices related research work is discussed. VM based resource scheduling algorithms are not studied. In Lucas-Simarro et al. (2013), a broker is designed for multiple clouds by using different VM strategies. In Kumar et al. (2019), scheduling scheme based resource provisioning and resource scheduling algorithms are studied. In Houssein et al. (2021) Singh et al. (2022), task scheduling algorithms along with meta-heuristic methods are emphasized. In Konjaang and Xu (2021), workflow scheduling algorithms are discussed in detail.

Fu et al. (2021) focused on scheduling problem in manufacturing systems. They have emphasized on swarm intelligence and evolutionary algorithm only. Sandhu (2021) presented comparative analysis of various cloud-based big data frameworks. However, the scheduling techniques implementing the cloud environment have not been studied in detail. The present survey categorized resource scheduling algorithms based upon problems addressed ; compared different scheduling techniques, and identified future research challenges in cloud computing environment.

The main objective of this paper is to provide a comprehensive survey of existing scheduling approaches. We want to convey the complexity involved at the time of scheduling, key taxonomy of scheduling methods and open issues and challenges.

1.1. Motivation

The main intent of the survey is to scrutinize the best techniques for the selection of appropriate resources for mapping of jobs for resource scheduling. This also includes stating the importance of cloud computing for accomplishing optimum resource utilization. A comparison of scheduling approaches using different IaaS providers is also discussed. It presents an exhaustive survey of cloud scheduling methods and compares those using different metrics. The following points are listed as those which motivated to conduct this survey:

- In a cloud computing environment, there is a requisite to understand the existing scheduling techniques.
- Resource scheduling in the cloud is a process of making scheduling decisions involving the allocation of jobs to ingredient resources. Consequently, we realized the need to emphasize cloud scheduling techniques by considering different scheduling criteria and different application requirements too.
- As cloud computing has emerged as a computing paradigm to provide the high-performance solution in comparison to the traditional supercomputing environment, diverse facets need to be considered to make effective cloud resource management systems. With its multitude of heterogeneous, geographically distributed, and dynamic nature of resources, resource provisioning and scheduling in the cloud is required for improving the performance of the cloud resource management system.
- To address the resource scheduling problem, it is mandatory to understand the requirement of applications. Cloud resource scheduling problems can be considered as a whole family of problems as there are many factors involved while addressing resource scheduling issues.
- A cloud service provider must provide the optimal solution for a highly dynamic geographical distributed environment, which emerges the intricate nature of efficient resource utilization and performance-driven solutions for application execution.

We identified the necessity of a methodical literature survey of cloud scheduling, as research in cloud resource scheduling is escalating these days. Therefore, we have summarized the available research based on broad and methodical search in existing archive and discussed issues and challenges for advanced research. An important issue here is how to formally choose an IaaS provider for scheduling problems in clouds. In this paper, we have presented the most important and useful aspects of scheduling in IaaS providers for this purpose.

1.2. Our contributions

The main intent of the research paper is to provide an overview of the scheduling concept and presents an exhaustive survey on cloud resource scheduling methods along with the importance of meta-heuristic methods. Our contributions are discussed as follows:

- A systematic survey of cloud resource scheduling methods along with merits and demerits.
- A taxonomy on resource scheduling algorithm is derived on the basis of both types of resources (Physical and logical resources) and resource scheduling scheme.
- Categorization and comparison of scheduling techniques have been done based on important characteristics and properties.
- Scheduling in different cloud service providers such as Amazon and Google Cloud is also discussed.
- A detailed study on optimization methods dedicated to optimization resource scheduling in cloud computation optimization is done.
- Analysis of challenges ahead and potential future research directions in the area of big data processing jobs, analytics as a service in the cloud are presented.

1.3. Organization

The rest of the survey article is organized as follows: Section 2 gives an overview of cloud resource management systems and the role of resource provisioning and scheduling in cloud computing. Section 3 presents the basic terminology of scheduling in cloud computing. It describes the basic characteristics of scheduling and phases of scheduling in cloud computing. Section 4 discusses the results of the systematic review. It also presents a taxonomy of scheduling in cloud computing systems that can be used to compare different scheduling algorithms on the basis of features and their analysis. Open issues and challenges for future work are also discussed in Section 5. Section 6 concludes the paper.

2. Cloud resource management systems: Background

This section outlines the basic concept of cloud computing along with its characteristics. The significance to manage the resources by keeping QoS parameters in consideration is presented. Few facts are also discussed to use cloud computing as a strategy of attaining the optimum cost and time for executing the application on clouds. The main role of resource provisioning and scheduling for the execution of the applications is also explained.

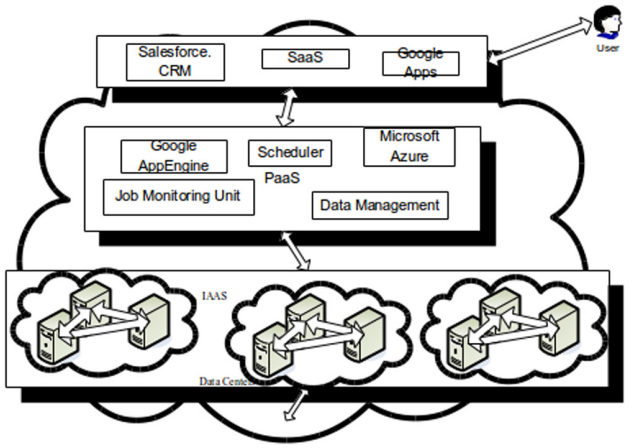


Fig. 1. Basic Cloud Architecture.

2.1. Cloud resource management systems

The term cloud computing originated and became the buzzword, driven largely by marketing and service offerings from big corporate players like Google, Microsoft, and Amazon. They offered on-demand access to infinite computing resources. The globalization of computing has now been made possible through clouds and the technology is now being offered in a pay-as-you-use service model (Rittinghouse and Ransome, 2009). Cloud services have been evolved as metered services (Foster et al., 2008).

In cloud computing, there are three service models and four deployment models that are used to provide the service to the users. The above definition makes cloud services distinct from traditional hosting by insisting on five key characteristics (1) on-demand self-service (2) dynamic and rapid elasticity (3) measured service (4) resource pooling (5) broad network access (Geelan et al., 2009; Mell and Grance, 2009). Based on the kind of services provided; three broad service models (as shown in Fig. 1) have been defined named as (1) Software-as-a-Service (SaaS) (2) Platform-as-a-Service (PaaS) and (3) Infrastructure-as-a-Service (IaaS). Analytics as a Service (AaaS) is also emerging as a service to provide the solution for big data processing jobs in cloud computing. It refers to the provision of analytic software and operations through web-delivered technologies. These solutions help in the development of hardware setup to provide business analytics. AaaS provides services for data analysis as IaaS offers computing resources. There are four deployment models named as (1) Private (2) Community (3) Public and (4) Hybrid.

Cloud computing has emerged as a much-awaited realization of a dream of transforming computing power into a utility that is commoditized and delivered as other existing utilities. It has the potential to transform the entire computing industry into a more attractive and easily accessible utility (Armbrust et al., 2010; Buyya et al., 2013). Capacity here is almost infinite, scalability is instantaneous, and consumers need to pay only for the resources they use as per the duration of use. In such an environment, there is a huge need to manage a large number of resources and providing ease of use to consumers as well.

Resource management system can be defined as a process of allocating the information system resources such as networking, storage, software, etc as per the requirement of application to meet the target and attain the better solution (Fox et al., 2009; Goyal and Dadizadeh, 2009; Jennings and Stadler, 2015). Efficient management of the dynamic nature of resource over geographical areas affect the performance of the deployment of the cloud for running applications. As there are many applications, hosted in the cloud environment to complete their task.

In cloud computing, there are different stages of resource management system such as resource provisioning, resource monitoring and jobs from job submission to job execution. In cloud resource management systems, we have considered mainly two stages: (i) resource provisioning and (ii) resource scheduling. Resource provisioning is defined as the stage to identify ingredient resources for a job based on the user's requirement for application execution whereas resource scheduling is the process to map jobs to appropriate resources so that the users get an optimal response.

2.2. Why do we choose clouds for application execution

Cloud computing has emerged as more optimized platform for the completion of application's task in an efficient way in comparison to the existing computing paradigms. In addition to provide the facility of on-demand resources and elastic services to different sectors such as e-commerce industry, scientific community; it also offers in the mapping of task to an ingredient resource.

The first objective of cloud scheduling is to identify the appropriate number of resources needed for the completion of a task as per the user's demand. For optimum scheduling of resources, an efficient resource provisioning strategy is required. The second objective is to meet the target of QoS parameters such as cost, time, resource availability, and resource utilization by allocating the proper resources to the jobs. For example, suppose a person has to order food in the food court. As per the requirement of food, the user will select the option of vegetarian or non-vegetarian by keeping the cost and time criteria into consideration (Rajni and Chana, 2013). However, the main point is that the implementation technologies of cloud computing make distributed computing a more optimized and efficient platform for the execution of the application.

Due to the facility of on-demand resources feature of cloud computing, we can use a huge number of resources for the execution of the application whenever required. Because of the multi-tenancy feature, cloud computing helps to enhance resource utilization and reduce energy consumption (Owusu and Pattinson, 2012). The maximum utilization of the resources support is provided by multi-tenancy and virtualization. Besides minimizing cost and time with the help of virtualization and multi-tenancy, cloud computing can be considered for the execution of different kind of applications using resource scheduling algorithms. Efficient resource scheduling algorithms support the process of minimizing real-time parameters, i.e., cost and time by keeping high resource utilization. The role of resource scheduling in clouds is discussed in the next Section 2.3.

2.3. Importance of cloud resource provisioning and scheduling

Cloud resource management systems consider as central environment for the execution of applications to attain high performance in a timely manner. Clouds can be handled in a more predictable way through a well-designed provisioning strategy. Resource provisioning and resource scheduling techniques are considered mandatory to attain high performance for the cloud computing environment.

Cloud resources are distributed over geographical, dynamic in nature, and modeling of the resources is done into the discrete units so that provisioning of resource could be done to the user in a proper manner (Rajni and Chana, 2010). There are QoS parameters associated with the application for the execution and to provide the guaranteeing QoS with resource provisioning in a timely manner is a challenging issue. Provisioning is a more complex problem in comparison to queuing process as the user has to select the sophisticated resource allocation decision (Joseph, 2009).

The process of resource scheduling aims to provide the facility of efficient resource usage by executing application. The main intent of the scheduling strategies is to fulfill the basic requirements of the services by considering specified QoS parameters. Consequently, the enhancement in resource utilization helps to reduce cost and time. In other words, the process of reducing time and cost for the application's

execution and the increasing levels of resource utilization (poor, average, higher) are not mutually exclusive occurrences. This is evident from the truth that if resources are over-utilized, it demands more time and cost. The state of resource under-utilization leads a node not to perform optimally, increase idle time, whereas an over-utilization state helps a resource to function in a proper way but sometimes it leads to degrade the performance of the resource too. Various resource scheduling algorithms, intent to enhance resource utilization simultaneously to reduce cost and time, are discussed in Section 5.

3. Cloud scheduling

3.1. Definition and basic terminology

Scheduling problems demand the task of allocating the resources to the jobs to attain objective function within specified constraints of resources. In classical scheduling problems, there is a need to define the sequence for allocating jobs on the resources as resources are fixed. In online scheduling, there is no need of future knowledge at the time of resource allocations. However, in cloud computing domain, service providers can add and release resources as per the requirement of user's application and thus control the number of resources being used to complete the task of application's execution. There are two types of scheduling exist in cloud computing: (i) client side: the mapping of task on VMs and (ii) provider side: the mapping of the virtual machines to physical machines. In this paper, we will study client-side scheduling in which the mapping of tasks on VMs is considered. We have described the scheduling terminology used in the cloud computing as follows:

Task: A task is a computation unit that will run in the cloud computing environment. As per the application's requirements, tasks are defined. A task can be independent and dependent. It cannot be further divided into smaller units.

Job: A job is designed by combining different tasks those having different processing capabilities. Every job may have different resource requirements too.

Workflow: A workflow is defined as a series of activities that need to be performed to execute the application.

VM: A virtual machine creates virtual environment with its own CPU, memory, storage and network on host computer system.

Cloud: Cloud can be defined as a model that helps to provide the services such as infrastructure, platform, software, data, analytics, etc to the clients on the basis of their requirements.

Provisioning: The formal allocation of the resources can be termed as provisioning.

Scheduling: The exact mapping of the job to an ingredient resource is called scheduling that aims to provide optimal solution.

Application: An application is designed to solve a large-scale problem in the cloud computing environment. An application consists of many jobs which require computations at the different level. An application can be considered as monolithic in which the whole application is assigned a complete computational node for the completion of the task.

Resource: A resource is considered as a computational unit that is used to complete the execution of the application's computational requirements. CPU, memory and network software, etc are the basic characteristics of the resource. It can be a source of information and expertise. A resource is a fundamental requirement for performing any task. There are different parameters associated with resources such as processing capabilities, data speed, storage space, and workload.

Schedulers: Schedulers are processes that decide which task and process should be accessed and run at what time by the available resources. It helps to keep the performance of the cloud at the highest level by scheduling in optimized way. Based on the frequency of schedule's operations, categorization is done: local scheduler, global scheduler and enterprise scheduler, etc.

Virtualization: The main task of virtualization is to create abstraction layer for running virtual instances. Virtualization simulated the hardware functionality.

3.2. Characteristics of scheduling

Efficient resource management, being the key to a successful cloud environment, banks hugely on a planned, effective, and efficient scheduling. Cloud scheduling still has to offer a tremendous scope of research due to its unique characteristics. While focusing on cloud scheduling, almost all its characteristics should be keenly observed and taken care of. Some of the obvious characteristics worth being aware of while focusing on cloud scheduling are discussed as below.

- Huge infrastructure: A cloud involving a plethora of resources spanning across the globe is obviously a huge structure. The range of tasks, jobs, and applications that need to get catered at any point of time too can be in large scale. To handle these small activities related to resources, an efficient resource management is required. Scheduling in this environment cannot be simple, one technique based, and fixed. A complex, dynamic and multi-faceted scheduling is the basic requirement to address the issues of compute-intensive and data-intensive applications.
- Scalable nature of clouds: Due to the dynamic nature of cloud, the resources belonging to different administrative domains keep on joining and leaving the system in a rapid manner. Being owned by different organizations, the resources offer minimal control over them.
- Resource heterogeneity: Resources in the cloud environment are highly diversified in nature, capacity, working style, and administrative domains.
- Highly diversified applications: There being no restriction on the source, nature, and purpose of applications being catered under a cloud environment, the presence of smart strategies is always expected. Jobs can be computation-intensive, data-intensive, a complete set of self contained applications, or an atomic task. A well-planned scheduling design can make a good balance among executions and fetch good solution.
- Diversity in resource connectivity techniques: There is no fixed technique to establish the connection with cloud resources.
- Decentralized resource ownership: In cloud computing environment, there is no single central resource provider.

3.3. Phases of scheduling

Al-khateeb et al. (2009) define cloud scheduling as the mapping of the job to the ingredient resources to get the optimal solution. Cloud scheduling involves two main stages: resource provisioning and scheduling.

- Resource provisioning: Resource provisioning means the formal allocation of resources in the cloud computing domain. There is a need to focus on other parameters such as application profiling and analogical benchmarking to provide the appropriate schedule. The allocation of the resources as per the specification given by user to particular application is called provisioning.
- Scheduling: Scheduling means the mapping of the task to the appropriate resources for the completion of the execution of applications.

3.4. Scheduling approaches

There are many different perspectives considered while studying cloud scheduling methods (Dong and Akl, 2006). Scheduling problems can be categorized as local vs global, static vs dynamic, centralized vs decentralized, co-operative vs non-cooperative, immediate vs batch, and approximation vs heuristic that are described as follows:

1. **Local versus Global**: In local scheduling, the main task is to select the process residing on a single CPU. In global scheduling, the information of resources is used to allocate the processes to different processors to get the optimal solution (Dong and Akl, 2006).

2. **Static versus Dynamic**: In static cloud scheduling, information about the resources and tasks is available at the time of scheduling. In dynamic scheduling, information is not available at the prior level. The basic information about the resource up-gradation and degradation will be keep posted (Li et al., 2011).
3. **Centralized versus Decentralized**: In centralized cloud scheduling, there is more responsibility on the cloud scheduler as the cloud scheduler having complete control over the resources. Cloud scheduling has to take best decision to manage the resources efficiently. There are many issues faced in centralized cloud scheduling problems, such as scalability, local balancing, and fault tolerance. In decentralized scheduling, there is no centralized control over the resources. The monitoring of the resources is done by the local scheduler (Xhafa and Abraham, 2010).
4. **Co-operative versus Non-cooperative**: In co-operative scheduling, each scheduling will have their own responsibility to manage their tasks to achieve the target. The mapping of the resources to a task is done as per the designed policy. In the non-cooperative cloud scheduling, each scheduler takes an independent decision for allocation of the resources and act as an autonomous entity (Dong and Akl, 2006).
5. **Immediate versus Batch**: In the immediate mode of scheduling, whenever a job will be entered into system, jobs will be scheduled immediately. In batch mode, tasks are grouped together to schedule for the execution of the application.
6. **Approximation versus Heuristics**: Approximation algorithms provide a worst-case performance guarantee in both computational time and solution quality, research on heuristics typically focuses on the average empirical behavior of the algorithms. In common scheduling problems, to get the satisfied solution, approximate algorithms are used. The usage of approximation algorithms is not possible as cloud resources are huge and dynamic in nature. The main reason not to use the approximation algorithm to solve cloud resource scheduling problems is that approximation algorithms are too slow for the large and dynamic scenarios considered in these problems. There is a need to focus on heuristic methods to solve large scale scheduling problems in distributed computing environments such as grid and cloud. Heuristic methods are considered as a de-facto standard for solving the grid scheduling issues as resources in the cloud environment are more dynamic in nature and distributed over the geographic areas (Vivekanandan et al., 2011).

4. A taxonomy of cloud resource scheduling algorithms

The main elements of cloud scheduling algorithms are discussed in this section. Basis, pros, cons, scheduling methods, experimental parameters, and performance matrices are compared with the existing scheduling methods. A detailed taxonomy is described as shown in Fig. 2.

4.1. Resource scheduling algorithms in cloud computing

In cloud computing, the level of resource utilization and scheduling are inter-related terms. Resource utilization has a great impact on the scheduling decision as the under-utilized resource consume more time and cost in comparison to proper-utilized resources.

In a cloud resource management system, resource provisioning and scheduling are co-related terms. After completing the resource provisioning, the scheduling is performed to achieve the optimal solution. Four main categories of resource scheduling algorithms are presented. A high-level taxonomy of resource scheduling algorithms is shown in Fig. 2. Table 1 shows the keyword used in google scholar to filter resource scheduling research papers.

Table 1
Keyword used in google scholar.

Problem related keyword used in problem	Scheduling related keyword	Heuristic based keyword
Cloud resource management system	resource scheduling	heuristic
Resource provisioning	Task scheduling	Meta-heuristic
QoS based resource scheduling	Workflow scheduling	Genetic algorithm based scheduling

4.1.1. VM based scheduling algorithm

Virtualization is an important technology which helps to achieve the goal of cloud computing. In VM scheduling, cloud service provider schedules the virtual machines on physical machines whenever user will request for the resources to execute the applications. In VM placement problem, the static part of the physical resources is shared among users (Jennings and Stadler, 2015). In this section, major types of VM placement are discussed as follows:

- *Dynamic VM placement:* In dynamic VM placement, the allocation and re-allocation of a virtual machine is done dynamically. The main aim of dynamic VM placement approach is to utilize the resource efficiently (Tighe et al., 2013). Dynamic VM placement is an important step to achieve system maintenance. To schedule resources across data centers, live migration is a more substantial method to improve fault management and to maintain the load balancing among physical machines too. Dynamic VM placement can enable a cloud service provider to achieve the main characteristics of cloud, i.e., elasticity. Tighe et al. (2013) presented a fully distributed approach for dynamic VM management. Their main concern is to decrease bandwidth consumption and enhance the power performance by removing centralized systems.
  Keller et al. (2014) investigated a hierarchical approach to data center management to improve scalability. They have created the hierarchy leveraging the data center network topology. In a hierarchical approach, they encapsulated VM migration and communication with the management scope. Tighe and Bauer (2014) have designed an algorithm for the automatic scaling of applications. They have considered the benefits from both perspectives, i.e., cloud client and provider.
  Sun et al. (2015) used the concept of online live migration for multiple correlated VMs. The Virtual Data Center (VDC) Migration (VDC-M) algorithm has been designed to solve the migration issue. After the process of remapping migration request, the task of computing the migration path is done. Finally, resource bandwidth is allocated to migrated VM.
  Tordsson et al. (2012) designed a multi-cloud resource provisioning architecture including a scheduling algorithm for the deployment of applications. The base of the scheduling algorithm is integer programming and other QoS related parameters such as budget and user's preference for selection of VM are also considered.
  Gutierrez-Garcia and Sim (2013) designed scheduling algorithm for bag-of-task applications. Agent based approach is used to design the scheduling algorithm. They have focused on concurrent, parallel selection of resources in the cloud environment. Table 2 shows the comparison of existing Dynamic VM placement based scheduling algorithm in cloud computing.
- *Energy aware VM placement:* Energy consumption is a major issue which is faced by cloud providers. By using server consolidation, optimizing operation on physical machines, and using dynamic voltage scaling processors, energy consumption can be reduced. Hu et al. (2010) presented a scheduling algorithm for load balancing of VM resources by using a genetic algorithm. The main intent of the algorithm is to provide information about the

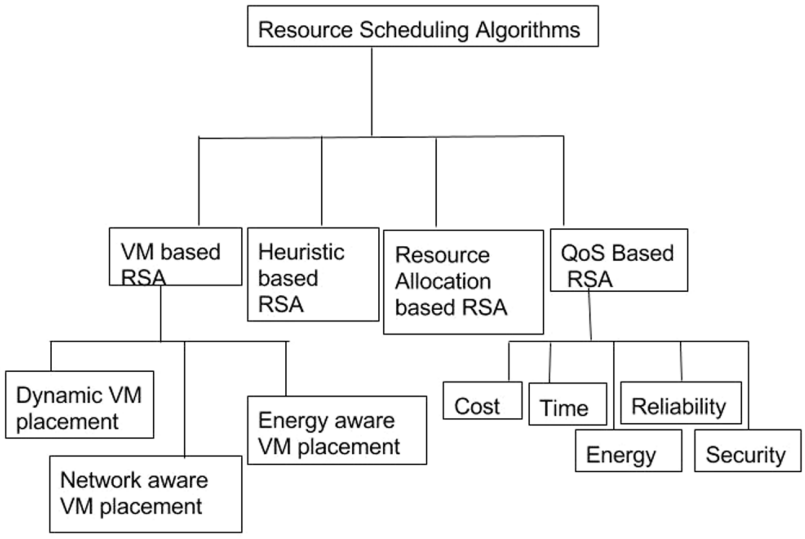


Fig. 2. Taxonomy of Resource Scheduling Algorithms in Clouds.

Table 2
Comparison of dynamic VM placement based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Tordsson et al. (2012)	Optimized placement of applications in multi-cloud environments.	Better Price and performance, Provided load balancing	Ignored security and energy efficiency at time of scheduling	Integer programming formulations	Amazon EC2	Throughput, number of jobs
Gutierrez-Garcia and Sim (2013)	Scheduling of Bag-of-tasks based on allocation times of virtualized cloud resources	Makespan	Ignored cost	Heuristic algorithm	Testbed	Makespan, overhead time
Tighe and Bauer (2014)	Proposed auto-scaling algorithm alongside a dynamic VM allocation algorithm	Reduce a number of migrations	No optimization criteria	Rule based heuristic	DC Sim	Power, SLA, Migrations
Tighe et al. (2013)	Focused on trade-off between VM migration and SLA violation	Consider energy and SLA	More bandwidth usage	First fit algorithm	DCSim	Power consumption, number of migrations, SLA violations
Keller et al. (2014)	To reduce the management scope by managing data center with the help of hosts	Reducing the overhead in the data center management network	High complexity	Greedy algorithm	DCSim	Power, number of migrations, average number of racks, active hosts
Sun et al. (2015)	Considered virtual data center to solve VM migration issues	Low complexity	Fixed bandwidth	Heuristic algorithm	Simulated environment	VM migration cost and time

resource on the basis of historical data and the current scenario of the system.

After completing this process, it selects the least effective solution, through which it achieves the main motive of algorithm "load balancing" and avoids dynamic migrations. This algorithm reduces high migration costs. Dabbagh et al. (2015) presented an integrated energy-aware resource provisioning framework for cloud data centers. The proposed resource provisioning framework caters to a prediction approach to predict both the number of VM requests and the number of cloud resources associated with each request. In this approach, authors have combined machine learning clustering and stochastic theory. Bui et al. (2016) designed an optimal energy-efficient architecture to orchestrate cloud systems. They have used prediction techniques to enhance the usefulness of monitoring statistics. They have tried to cut the running physical machines by stack VM.

A decentralized architecture of the energy-aware resource management system for cloud data center is presented in Beloglazov and Buyya (2010). They have introduced an approach for allocation of VMs in run-time by applying live migration. After getting the information of the current utilization of resources, the authors have done live migrations. The main intent of the authors is to minimize energy consumption by attaining quality of services. A comparison of existing energy-aware VM placement-based scheduling algorithms in cloud computing is shown in Table 3.

- *Network aware VM placement:* An ever-increasing number of customers for executing the application on cloud impact the proliferation of network traffic. It has become essential to share the network bandwidth without hurting the VMs quality of service (Breitgand and Epstein, 2012). In the cloud computing environment, to establish communication with other application

Table 3
Comparison of energy aware VM placement based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Hu et al. (2010)	Focused on vm resources for load balancing	Reduced migration cost and help to improve load imbalance	Ignored security and time	Genetic algorithm	Open nebula	Number of migrations, migration cost, load of physical machine
Beloglazov and Buyya (2010)	Optimization of systems resource by monitoring currently available resources such as CPU, ram and network etc.	Do not depend on a particular type of workload and No prior information about applications executing on VMs	Ignored cost and time	Heuristic algorithm	CloudSim	Energy, Average SLA, migrations
Dabbagh et al. (2015)	Presented energy-aware resource management decisions	Improved performance	No optimization criteria, high complexity	K-means clustering	Testbed	Average CPU and Network utilization
Bui et al. (2016)	To maintain balance between energy efficiency and quality of service	Low complexity	Ignored cost, scalability	Greedy first fit algorithm	Simulated Environment	Energy, Memory, CPU

and system components, VMs access a PM's network interface. Hypervisors typically treat a PM's resource such as network interfaces, switch links as un-managed resources. Hypervisors do not provide a guaranteed allocation to individual VMs. It is relying on statistical multiplexing and the fact that VMs are unlikely to simultaneously maximize the use of their nominally assigned bandwidth. However, this means that there is potential for VMs to affect each other's performance due to contention for network interface resources. A stable traffic-aware VM placement study is discussed in Biran et al. (2012). They have proposed Min Cut Ratio-aware VM placement algorithm to solve the placement solutions resilient to traffic bursts in deployed services, by minimizing the maximum load ratio over all the network cuts. Yu et al. (2017) proposed a novel dynamic programming-based algorithm to embed virtual clusters survivable in the cloud data center by optimizing virtual clusters. They have computed the most resource-efficient embedding given a tenant request.

Li et al. (2021) designed a reinforcement learning based scheduling algorithm by analyzing different scheduling schemes for radio resources using the Vienna 5G SL simulator. Kondikoppa et al. (2012) deployed global MapReduce over federated clusters to utilize the computing effectively. They have provided network awareness to the FIFO and FAIR schedulers in Hadoop. An extended architecture for NFV and network-aware scheduling in OpenStack is presented in Lucrezia et al. (2015). The authors have presented the extension of network-aware scheduler to optimize the VM placement. This step is an essential part of deploying virtual network functions service graphs. Rampersaud and Grosu (2016) developed sharing-aware online algorithms for addressing VM Packing issues. Two strategies best fit and worst fit are used to define resource scarcity metric in the paper. Table 4 shows the comparison of existing network-aware VM placement based scheduling algorithms in cloud computing.

4.1.2. QoS parameter(s) based scheduling algorithm

In this section, we present the scheduling algorithm with the Quality of Service (QoS) requirements of applications. The main focus of QoS parameter(s) based scheduling algorithms is to not only reduce real-life based factors, i.e., cost and time, but also to help in the decision process of scheduling in the cloud resource management systems. A cloud resource management system needs to satisfy the quality requirements of an application. It is very important to figure out the real stakeholders to list down the requirements of the system. A categorization of stakeholders along with quality requirements is shown in Table 4. For instance, cloud service provider wants to maximize profit by ensuring that all user requirements to execute an application is fulfilled by proper utilization of resources. Cloud service providers are also worried about the operation costs which occur due to unnecessary executions. To operate high-performance system, energy consumption is increasing at a high pace and it also leads to failure of the system. Cloud users who submit jobs to the cloud resource management system expect response time and cost to be minimum. They want to execute their job in a secure environment by fulfilling SLA requirements.

Table 5 shows the quality requirements for designing scheduling algorithms under QoS based resource scheduling algorithms. Resource scheduling algorithms are designed to achieve the target set by the user to get the optimal solution to the scheduling problem. With the help of one scheduling algorithm, it is really difficult to meet the different quality of requirement at the same time. For instance, the well-known deadline aware scheduling algorithms only concentrate on mapping of the job to an ingredient resource to provide the response of the application before time and user requirements. In this section, we study resource scheduling algorithms from the perspective of fulfilling the quality requirements. We defined the most common quality attributes considered at the time of designing scheduling algorithms.

1. *Cost based scheduling algorithm:* Cost aware resource scheduling algorithms play an important role in cloud resource management systems as the main intent of clouds is to provide the resources on-demand and work on pay-as-you-go principle.

   Yuan et al. (2017a) proposed a scheduling algorithm by keeping the aim to increase profit and by giving guaranteeing service delay bound of delay-tolerant tasks. To solve the task scheduling problem, the author has focused on the heuristic algorithm PSO and SA. In Yuan et al. (2016), authors have focused on cost-aware load scheduling to achieve high throughput at less price. Yuan et al. (2017b) have designed an algorithm to minimize the cost. In Yuan et al. (2018) research paper, the authors have proposed workload-aware revenue maximization algorithm to schedule the application in software defined networking enables data centers. They have not only focused to reduce the network latency but also considered virtual machine latency too (Ghahramani et al., 2017).

   Bi et al. (2015) proposed architecture for multi-tier web applications that helps in the self-management of the data centers. To identify the total number of virtual machines required for the execution of an application, a hybrid queuing model is designed. In Bi et al. (2016), authors have designed a scheduling algorithm for temporal requests. They have seriously considered the delay parameters. A cloud task scheduling framework is presented by using two stage strategy. They have focused on QoS parameters cost and deadline factors (Zhang and Zhou, 2018).

Table 4
Comparison of network aware VM placement based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Yu et al. (2017)	Service provisioning in an IaaS cloud environment	To achieve the high availability goal of tenant services	High complexity	Heuristic algorithm	Simulator	Average VM consumption ratio, average running time
Kondikoppa et al. (2012)	Designed Hadoop scheduler aware of network topology	Improved data locality	Ignored cost, energy, security	FIFO	Eucalyptus based testbed	Execution time, delay for scheduling task
Lucrezia et al. (2015)	Investigated to what extent OpenStack needs to be modified to support the deployment of network service graphs	Increased throughput	Analyzing time is more, Ignored policy-constraints in order to define administration rules	Brute force algorithm	KVM hypervisors	VM locations, traffic throughput and latency
Biran et al. (2012)	Consideration of traffic bursts in deployed services	Minimizing the maximum load ratio over all the network	Ignored energy consumption	Greedy heuristic algorithm	Testbed	Average packet delivery delay, placement solving time
Rampersaud and Grosu (2016)	Focused on VM packing problem	Consideration of the potential for memory sharing when making allocation decisions	High complexity	Linear programming technique	Simulated environment	Memory reduction, number of excess servers

Table 5
Quality of service requirements.

Stakeholder	QoS requirements
Cloud service providers	Operation cost, Time, Profit, Energy efficiency, Reliability
Cloud users	Execution cost, response time, security

Arabnejad and Bubendorfer (2015) proposed a Proportional Deadline Constrained (PDC) algorithm, for workflow scheduling in the cloud computing environment. The main intent of the proposed algorithm is to reduce cost by targeting deadline constraints. A Budget Distribution with Trickling (BDT) algorithm is presented to handle the trade-off between cost and time for the workflow scheduling (Arabnejad et al., 2016). The authors have used already provisioned resources to reduce the execution time for the workflow scheduling.

Zuo et al. (2015) proposed a resource cost model that can be used to define the task demand for resources in detail. They have used Ant Colony Optimization (ACO) algorithm to design the model. The main concern was to optimize the scheduling of performance and user costs.

Wu et al. (2012) proposed scheduling methods to address admission control services. They considered multi-objective resource provisioning strategies. The objectives of the proposed work are to minimize resource consumption and penalty costs as well as minimizing SLA violations.

Ari and Muhtaroglu (2013) designed a smart scheduler that can dynamically select some of the required parameters, partition the load and schedule it in a resource-aware manner. Lee et al. (2012) developed a pricing model using processor-sharing for clouds. They have proposed algorithms for scheduling service requests and prioritizing their data accesses in the cloud with the main aim of maximizing profit. A three-tier cloud structure, which is a representative cloud system model has been adopted in Lee et al. (2012). Mangla et al. (2021) discussed scheduling techniques to provide the optimal solution. Their main intent is to achieve maximum utilization and user satisfaction. Table 6 presents a comparison of cost-based scheduling algorithms.

2. *Time based scheduling algorithm:* Yuan et al. (2017c) proposed a time-aware task scheduling algorithm in green data centers. The authors have used a meta-heuristic algorithm by combining particle swarm optimization and simulated annealing to design scheduling algorithm. The algorithm helps to investigate temporal variations. The algorithm is used to consider delay bounds. Khojasteh Toussi and Naghibzadeh (2021) and Priya et al. (2019) considered the multi-dimensional resource allocation problem. They have proposed a scheduling algorithm to achieve the main target of load balancing. Arabnejad et al. (2017) considered the problem of deadline constrained scientific workload scheduling. In this paper, proportional deadline constrained and Deadline Constrained Critical Path (DCCP) is designed to provide the solution of workflow scheduling. To provide the facility of task communication on the same instance, critical paths are determined.

   Thomas et al. (2015) introduced user priority-based scheduling algorithm. The main focus of the improved algorithm is to consider cloud user requirements and resources obtain-ability. They have used min–min method to reduce the makespan. A data-aware scheduling algorithm was designed for cloud providers by in Van den Bossche et al. (2013). The authors proposed scheduling algorithms for deadline-constrained bag-of-tasks applications while considering data locality. They have focused on task runtime discrepancies. In this scheduling technique, a static set of applications has been considered for scheduling on both public and private cloud infrastructure.

   Xu et al. (2011) proposed Berger model-based job scheduling algorithm. The main base of the Berger model is the distribution theories of social wealth. The algorithm focused on dual fairness constraints: (i) On the basis of QoS preferences, user's task classification and to provide the fairness among resource selection while meeting expectations and (ii) to define resource fairness fitness function for the evaluation of the resources allocation scheme.

   Frîncu (2014) presented scheduling algorithms for achieving high availability. They have used the concept of component-based architecture. Erdil (2013) proposed proxies-based scheduling algorithm. Information about resources is available with the help of proxies. To provide an optimal solution, a bottom-up approach is used. Table 7 presents a comparison of time-based scheduling algorithms.

3. *Energy based scheduling algorithm:* Energy-aware resource scheduling helps to extract the total amount of energy consumed for the processing of an application, depending upon the ingredient resource for executing jobs. There is a need to focus on energy-aware resource scheduling methods as these methods

Table 6
Comparison of cost based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Yuan et al. (2017a,b)	Proposed scheduling algorithm by emphasizing on profit maximization	Handles service delay bound	High complexity	PSO and SA	Simulation environment	Revenue
Bi et al. (2015, 2016)	Designed an architecture for self management of data centers	Considered temporal request of multi-tier web applications	Does not consider security parameters	Queuing approach	Trace-driven simulation	Cost
Wu et al. (2012)	Focused on VM placement issue on the basis of time	Up to 40% cost saving	Does not support security and energy efficient	Admission control and scheduling algorithm	CloudSim	Average response time, total profit
Ari and Muhtaroglu (2013)	Discussed finite Element Analysis cloud services and job scheduling issues	Throughput improvement and resource utilization	Ignored cost	Adaptive algorithm	Testbed	Throughput and time
Lee et al. (2012)	Presented the personalized features of the user request and the elasticity of SLA properties	Reduced operational costs and increase profits	Objectives conflict with each other	Binary integer programming	CloudSim	Average utilization, average net profit rate, average response time
Mangla et al. (2021)	Focused on scheduling techniques	Reduce cost and time	Ignored reliability and security	Optimum scheduling algorithm	CloudSim	Cost and Time
Arabnejad and Bubendorfer (2015)	Focused to re-use of pre-provisioned instances for scheduling	Less complexity	Ignored security and energy efficiency	Deadline early Tree algorithm	CloudSim	Cost and deadline
Zuo et al. (2015)	Multi-objective Task Scheduling	Improved performance	Ignored energy consumption	Ant colony optimization	CloudSim	Cost, makespan, deadline violation rate

play important role in the data centers to reduce energy consumption (Calheiros and Buyya, 2014). Data is increasing at a very high speed and there is a need to process the data by server and disks in the specified time frame. Due to above-mentioned reason, there is a huge wastage of idle power that increases energy consumption (Beloglazov et al., 2012). Bessis et al. (2013) have developed a model for efficient message exchange systems for distributed systems. Their main aim is to reduce energy consumption and total makespan.

Bi et al. (2017) have proposed a dynamic meta-heuristic algorithm for scheduling in virtualized data centers. Their main aim is to enhance energy efficiency and increase profits by fulfilling SLA conditions. Zhu et al. (2017) proposed dynamic power resource model of the cloud data centers. Based on three-dimensional CPU, RAM, and bandwidth, a virtual resource (VR) scheduling method is also developed to save energy. Moreover, they have considered how to keep balance among cloud data centers. An efficient prediction model is designed for energy-aware scheduling, by Duan et al. (2017). The base of the model is fractal mathematics. The authors tried to handle the energy consumption during peak hours in heterogeneous clouds.

Quarati et al. (2013) presented a cloud brokering algorithm. QoS parameters energy consumption, revenue, and user satisfaction are considered at the time of scheduling. A cloud-aware scheduling algorithm has been proposed by Calheiros and Buyya (2014). Moreover, their main concern is to use the Dynamic Voltage and Frequency Scaling (DVFS) factor to maintain the voltage level of CPU processors at the time of scheduling. Deadline constraint is considered for scheduling of bag-of-tasks. A new VM scheduler has been developed by Ding et al. (2015). Their main focus to reduce energy costs for the scheduling of applications by fulfilling deadline constraints. Li et al. (2016) used heuristic method to provide energy based scheduling algorithm solution for workflow applications.

Kim et al. (2014) presented a model to predict the energy consumption of each virtual machine. This model works on the basis of in-processor events generated by the VM. This model incorporated the scheduling algorithm for mapping the tasks to the resource under the energy constraint. Van Do and Rotter (2012) designed an analytical performance model for handling energy efficiency requests at the time of scheduling of applications. Interactions between cloud users and service providers are focused to fix energy issues.

Garg et al. (2011) designed scheduling algorithm to address the energy consumption issues for cloud service providers. A cloudlet-based mobile cloud computing model (DECM) has been presented to solve the issue of energy consumption (Gai et al., 2016). A comparison of existing energy-aware based scheduling algorithms with respect to scheduling parameters is done in Table 8.

4. *Reliability based scheduling algorithm:* Malik et al. (2012) designed a reliability assessment model for cloud infrastructure. Fault tolerance parameter is handled by checking the reliability of nodes in this model at the time of scheduling of applications.

Adhikari et al. (2020) designed scheduling algorithm for a workflow using the firefly approach. Their main intent is to reduce makespan while achieving reliability.

Jing et al. (2015) have introduced a model to handle fault-tolerant scheduling issues by considering the communication of network interfaces along with processor resources. To improve reliability, authors have used the replication mechanism and reliability estimation.

Latiff et al. (2016) proposed a Dynamic Clustering League Championship Algorithm (DCLCA) scheduling method. The main function of the proposed method is to provide the solution of load balancing by considering fault tolerant nodes. These techniques have emphasized the available resources and handle the failure of independent tasks.

Tang and Tan (2016) introduced a reliability and energy-aware task scheduling architecture. In this model, parallel applications are considered while addressing energy consumption as QoS parameter. The single processor failure rate model has

Table 7
Comparison of time based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Priya et al. (2019)	Considered load balancing time at the time of scheduling	Reduce latency time	Ignored security and High complexity	Fuzzy Method	Simulated Environment	Response time
Thomas et al. (2015)	Focused on task length aware scheduling	Lesser makespan and increased resource utilization	No comparison with existing algorithm	Min–min	Cloud Sim	Makespan
Van den Bossche et al. (2013)	Presented migration of VMs as desired to adjust to varying performance needs, scheduling algorithms for deadline-based workloads in a hybrid cloud setting	Minimize cost and time	Does not handle multiple workflows	Hybrid scheduling approach	Simulator	Total Cost, application deadline met, turnaround time, data transferred
Yuan et al. (2017c)	Concentrated on task scheduling in green data centers	Investigate temporal variations	Ignored energy consumption and cost	PSO and SA	Simulated Environment	Delay bound and time
Arabnejad et al. (2017)	Addressed the problem of workflow scheduling in dynamically provisioned commercial cloud environments.	Evaluation of task selection algorithms reveals impact of workflow symmetry	High complexity	Rank method	CloudSim	Response time, Cost
Xu et al. (2011)	Worked on Berger model and assign tasks on optimal resources to meet user's qos requirements	Optimal completion time	Ignored cost and energy efficiency, security	Resource allocation algorithm and then followed by job scheduling	CloudSim	Time, bandwidth
Frîncu (2014)	Considered the method of scaling to be known a priori and focus on searching for an optimal allocation of components on nodes in order to ensure a homogeneous spread of component types on every node.	Minimizing the application cost	Centralized approach represents a single point of failure	Nonlinear-programming	Simulator platform	Average load per node, optimal allocation, reliability
Erdil (2013)	Disseminated information as agents of dissemination sources for resource scheduling	Availability of resource state, reduces dissemination overhead	Ignored cost as parameters	Adaptive proxy algorithm	Scalable simulation network framework	Query satisfaction rates, random walk hop count limit

been proposed to address the reliability issues of applications. Reliability-Energy aware scheduling heuristic method has been designed to execute the application with an aim to get trade-off among reliability and energy consumption. A comparison of existing reliability based scheduling algorithms with respect to scheduling parameters is done in Table 9.

5. *Security based scheduling algorithm:* In this section, security-aware scheduling algorithms have been presented as security is one of the major constraint to be satisfied by the scheduling algorithm. In a security-aware cloud computing environment, the main responsibility of the VM scheduler is to provide security along with minimizing the response time of the application's execution simultaneously (Kashyap and Vidyarthi, 2014).
Chejerla and Madria (2017) proposed a game theory-based scheduling method for application's execution in the cloud computing environment. They have used Bayesian network to design an algorithm. While scheduling, they took the decision on the basis of output that comes from the game. Through a new method, authors able to design a security-aware scheduling algorithm. A novel VM placement algorithm is designed to handle the security issues (Shetty et al., 2016). They have discussed VM placement strategies and VM vulnerabilities to address security concerns at the time of scheduling.
Zeng et al. (2015) designed a security-aware and budget-aware scheduling method for workflow applications. The main intent of the author is to focus on communication-intensive workflows by considering different types of datasets. Security, resource utilization, and budget requirements as QoS requirements are considered. Kashyap and Vidyarthi (2014) proposed a new credit allocation policy for solving the dual objective scheduling problem. They have designed a real-time security maximization scheme to fix deadline constraints along with security issues for VM allocation.
A novel authenticated key exchange scheme is designed to facilitate security to the scientific applications in the cloud computing environment (Liu et al., 2013). The base of the scheme is the randomness-reuse strategy and the Internet Key Exchange (IKE) scheme. Wang et al. (2012) designed a cognitive trust model by incorporating a dynamic level scheduling algorithm. The main function of the model is to reduce the failure probability of the task assignments.
Afoulki et al. (2011) proposed a security-aware scheduler for cloud environment. They have embedded these policies in VM placement and migration algorithms. The proposed scheduler will place VM's on the same PC's by avoiding security risk. Bilogrevic et al. (2011) have presented a privacy-preserving method to address server scheduling problems. To compute common user availabilities, the authors have used the concept of homomorphic properties of well-known crypto-systems. Duan et al. (2019)

Table 8
Comparison of energy based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Bi et al. (2017)	Dynamic Scheduling algorithm for reducing energy consumption's	Focused on performance and energy cost	High complexity due to virtualized Data centers	Meta heuristic methods	Simulated environment	Profit, CPU utilization
Zhu et al. (2017)	Focused to keep data center in balance stage while saving power consumption	Improved management of VM resource	High complexity	Multi-dimensional vector bin packing problem based heuristic	CloudSim	SLA violations, resource utilization
Duan et al. (2017)	Scheduling of VM machines	Improved the CPU load prediction	No optimization	Ant colony optimization	CloudSim	Energy Consumption
Ding et al. (2015)	Dynamic VMs scheduling	Increased Processing Capacity	Ignored VM migration, power penalties of status transitions of processor	FCFS	Simulated environment	Deadline, Energy consumption
Calheiros and Buyya (2014)	Exploited intelligent scheduling combined DVFS capability	Improved energy efficiency	Ignored Network and Storage energy consumption	Rank method	Cloud Sim	Energy consumption
Quarati et al. (2013)	Emphasized on the reservation of a quota of private resources	Reduced energy consumption and carbon emission	Lacks implementation on a real-world cloud platform	Round robin algorithm	Discrete Event Simulator	User satisfaction, energy saving, energy consumption
Kim et al. (2014)	VM energy consumption estimation model	Reduced cost, power consumption	More complex to implement, Ignored time	Power aware scheduling algorithm	Xen 4.0 hypervisor	Energy consumption, error rate
Van Do and Rotter (2012)	Presented interaction aspects between on-demand requests and the allocation of virtual machines	Reduced energy consumption	No cost and time optimization	Power aware scheduling algorithm	Numerical Simulation	Average Energy consumption, average heat emission
Garg et al. (2011)	Focused on optimal scheduling policies	Reduced energy cost, energy consumption	Ignored security	Meta-scheduling policies	Simulated environment	Average energy consumption, average carbon emission, arrival rate of application
Gai et al. (2016)	Considered functionality of cloudlets for energy reduction	Reduced energy consumption	No time consideration	FCFS scheduling policy	DECM-Sim	Energy consumption
Li et al. (2016)	Presented scheduling algorithm to reduce energy consumption while meeting the deadline constraint	Focused on energy consumption	Ignored processing power energy consumption, VM migration	Heuristic method	Simulated environment	Energy consumption
Bessis et al. (2013)	Focused on improving communication for Distributed systems while scheduling	Improved system performance	High complexity	Graph theory concepts	SIMIC	Makespan, latency times

presented a scheduling algorithm to attain the target of energy efficiency by providing security. A comparison of existing security based scheduling algorithms in cloud computing is presented in Table 10.

4.1.3. Heuristic based scheduling algorithms

Most of the aforementioned QoS parameters based scheduling algorithms have focused on providing efficient mapping of resources at the internal logical level. For solving NP-complete scheduling problems, there is a need to use heuristic methods. Heuristic methods help to design efficient scheduling plans to fulfill the requirements as per user's applications. Pandey et al. (2010) presented a Particle Swarm Optimization (PSO) based heuristic method for scheduling of the application in the cloud computing environment. Computation cost and data transmission cost are considered at the time of scheduling of applications. Akbar and Irohara (2020) focused on dual constrained resource allocation problems. They have emphasized the multi-task scheduling problem. Gąsior and Seredyński (2016) proposed a parallel and distributed scheduling method in cloud computing environment. Two methods such as multi-objective genetic algorithm with Spatial Prisoner's Dilemma game and the Sandpile CA model are used to provide the optimal solution by minimizing job completion time. Mateos et al. (2013) proposed a cloud scheduler based on bio-inspired techniques like ant colony optimization and swarm intelligence for parameter sweep experiments. The scheduler considers job priority. They have minimized flowtime and makespan as QoS parameters for independent job/s execution. Kousiouris et al. (2011) suggested a black-box method to handle the degradation prediction ability of resource. The proposed method is based on artificial neural network method. A comparison of heuristic methods based scheduling algorithms is done in Table 11.

Torabzadeh and Zandieh (2010) proposed a cloud theory-based simulated annealing solution for two-stage assembly flow-shop problems where there are m number of machines in the first stage and an assembly machine in the second stage. Mezmaz et al. (2011) designed a new parallel bi-objective hybrid genetic algorithm for the scheduling of parallel applications that takes into account, not only makespan

Table 9
Comparison of reliability based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Latiff et al. (2016)	Focus on uncountable numeric nodes for resource in clouds	Lower makespan	No optimization	League championship algorithm	CloudSim	Failure ratio, the failure slowdown and the performance improvement rate
Tang and Tan (2016)	Reliability and energy aware task scheduling architecture	To get good trade off among performance, reliability, and energy consumption	No support for cost optimization	Heuristic method	Discrete event simulation environment	Schedule length, Energy consumption, Application reliability.
Adhikari et al. (2020)	Focused on workflow scheduling	Focused on the enhancement of resource utilization	Reliability and Ignored security and Cost	Firefly Method	Simulation Environment	Reliability, Makespan
Malik et al. (2012)	A reliability assessment mechanism for scheduling resources	Reliability assessment algorithms for general applications and real time applications.	No security and energy parameters consideration	Max–min	Amazon EC2 cloud	Fault tolerance, time
Jing et al. (2015)	A model for fault-tolerant aware scheduling	Low complexity	No cost, time optimization	Adaptive secure scheduling algorithm	Simulated environment	Reliability

Table 10
Comparison of security based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Duan et al. (2019)	Focused on secure scheduling	Data privacy	Complex to implement	Security based algorithm	Simulation environment	Energy, Security
Kashyap and Vidyarthi (2014)	Secure aware scheduling of real time based applications	Improved response time and overall security	High complexity	Priority Algorithm	Hypervisor	Deadline, security
Liu et al. (2013)	Scheme for security aware scheduling	Reduced the computational load and execution time	No cost optimization involved	Adaptive secure scheduling algorithm	KVM hyper-visor	Time unit consumed per computational load
Wang et al. (2012)	Focus on uncountable numeric nodes for resource in clouds	Provided scheduling of resources in secure way	Ignored cost	Bayesian algorithm	CloudSim	Trust value, average schedule length
Afoulki et al. (2011)	Handled security risks with in cloud	Less complexity	Consolidation issues while implementing policies	Greedy Algorithm	Simulated environment	VM placement time
Bilogrevic et al. (2011)	Scheduling services on the cloud for mobile devices	Enhance Performance	No support cost optimization, Ignores power consumption by the network	Privacy aware scheduling schema	Testbed	Time, Data exchanged, privacy in approach
Zeng et al. (2015)	To provide robust scheduling algorithm for resource utilization	Low complexity	Ignored energy consumption	Clustering and prioritization algorithm	Simulated environment	Makespan and speed up
Chejerla and Madria (2017)	Scheduling of resources in cloud integrated Cyber–physical Systems	Consideration of security, time	High complexity	Heuristic algorithm	Simulated environment	Speed up, resource utilization, makespan
Shetty et al. (2016)	Considered VM placement techniques to reduce security risks	Reduced computing costs and deployment costs	No optimization criteria	Heuristic algorithm	Simulated environment	Cost, security risks

but also energy consumption. They have investigated the problem by considering makespan and energy consumption as QoS parameters.

Su et al. (2013) presented a cost-efficient task-scheduling algorithm. Two heuristic methods are used for the mapping of the tasks on the ingredient resources. In the first heuristic method, Pareto dominance is used to mapping the task on VM by considering cost as an important factor. In the second heuristic method, authors tried to reduce the monetary costs and makespan of non-critical tasks simultaneously. Abrishami et al. (2013) proposed workflow scheduling algorithms named IaaS cloud Partial Critical Paths (IC-PCP), and IaaS cloud Partial Critical Paths with Deadline Distribution (IC-PCPD2). IC-PCPD2 is designed by enhancing the Partial Critical Path (PCP) algorithm. The main difference between these two algorithms is that IC-PCP handles the scheduling process in one phase by selecting once the critical path for mapping the tasks on a single instance and IC-PCPD2 assign deadlines to the tasks and then assign tasks to the resources on the basis of sub deadline. QoS parameters such as deadline and cost are considered at the time of scheduling.

LD and Krishna (2013) have used Artificial Bee Colony (ABC) for load balancing algorithm in cloud computing environments. This algorithm has taken into the consideration the priorities of tasks. They have only considered priority as QoS parameters. Bousselmi et al. (2016)

Table 11
Comparison of heuristic based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Akbar and Irohara (2020)	Focused on resource allocation problem	Improved time	High complexity	Simulation environment	Tabu search	Time
Kousiouris et al. (2011)	Virtual machines affect the performance of other VMs executing on the same node	Reduce performance overhead	Lacks implementation on a real-world cloud platform	Genetic algorithm	Simulated environment	Degradation, test score delay
Abrishami et al. (2013)	Cost-optimized, deadline-constrained execution of workflows in cloud. Considered required runtime and data estimates in order to optimize workflow execution	Minimize execution cost with in deadline	Ignored data transfer time, security	PCP algorithm	Simulated environment	Normalized cost
Torabzadeh and Zandieh (2010)	Considered flowshow job problem	Minimized makespan and mean completion time	Not considered cost	Simulated annealing	Simulated environment	Computation time
Mezmaz et al. (2011)	Addressed the precedence-constrained parallel applications for cloud computing.	Reduced energy consumption	High complexity of implementation and operation	Genetic algorithm	Simulated environment	Energy, speed up
Su et al. (2013)	Cost-efficient task-scheduling algorithm using two heuristic strategies	Reduced monetary costs	Ignored security	Heuristic strategies	Numerical experiments	Makespan
LD and Krishna (2013)	Based priority of tasks, designed load balancing algorithm	Maximize throughput	High operational complexity	Honey Bee algorithm	CloudSim	Makespan, Number of task migrations
Bousselmi et al. (2016)	Designed scheduling algorithm	Consideration of QoS parameters	High complexity	Parallel Cat Swarm Optimization	Simulated environment	Execution time, execution and storage cost, availability of resources and data transmission time
Devi and Valli (2021)	Focused on resource scheduling with the help of workload prediction	Considered cost	Ignored Time and security	Genetic algorithm	Simulation environment	Cost, Resource utilization
Gąsior and Seredyński (2016)	A novel parallel and distributed scheme for scheduling jobs	Multi-objective optimization, consideration of security risks also	No cost consideration	Genetic algorithm	Simulation Testbed	Flow time, makespan, turnaround time
Mateos et al. (2013)	Designed scheduler for job scheduling, consider static cloud	Minimize weighted flowtime and makespan	Does not handle energy consumption	Ant colony optimization and swarm intelligence approach	CloudSim	Makespan

used a Parallel Cat Swarm Optimization (PCSO) heuristic method for scheduling the scientific workflows. The main intent of the authors is to reduce execution time, cost and to handle data placement issues simultaneously at the time of scheduling. Devi and Valli (2021) designed scheduling algorithm by calculating the total number of virtual machines at the advance stage. They have used a genetic algorithm to address the scheduling issue. Hu and Li (2021) proposed improved heuristic job scheduling method. The method named Densest-Job-Set-First (DJSF) method schedules jobs by maximizing the number of completed jobs per unit time, aiming to decrease the average Job Completion Time (JCT) and improve the system throughput. Attiya et al. (2022) designed a novel hybrid swarm intelligence method, using a modified Manta-Ray Foraging Optimizer (MRFO) and the Salp Swarm Algorithm (SSA) for scheduling IoT applications on Cloud.

4.1.4. Resource allocation based scheduling algorithms

In this section, resource allocation based scheduling algorithms are discussed. To maximize the under-utilized resources, the resource allocation step needs to be performed prior to the scheduling step. Nathani et al. (2012) proposed scheduling algorithm based on dynamic planning. The proposed scheduling algorithm is implemented on the Haizea framework. In the Haizea framework, the lease approach is used. Whenever, there is a need to start a new VM for application execution, a new lease is adopted. Four different policies such as immediate, best effort, advanced reservation, and deadline sensitive are used in Haizea framework to provide an efficient resource allocation for the scheduling.

Ma et al. (2019) designed resource scheduling algorithm to minimize the cost for smart grids. They have used the concept of cooperative learning. A new priority-based job scheduling algorithm (PJSC) in cloud computing has been proposed by Ghanbari and Othman (2012). The base of the proposed algorithm is decision making model that depends on multiple criteria. Li et al. (2012) proposed online dynamic resource allocation algorithms in the federated heterogeneous cloud systems for preempt-able applications. Mateescu et al. (2011) have introduced a hybrid High-Performance Computing (HPC) infrastructure architecture. The main function of the HPC infrastructure architecture is to utilize the resource provided by different cloud service providers for the execution of scientific applications.

Table 12
Comparison of resource allocation based scheduling algorithms.

Author	Basis	Merits	Demerits	Scheduling method	Experimental environment	Performance metrics
Nathani et al. (2012)	Proposed dynamic planning based scheduling algorithm that can admit new leases and prepare the schedule whenever a new lease can be accommodated	Offers deadline based optimization	Increased complexity for implementation	Backfilling and swapping techniques	Haizea Simulator	Number of deadline leased, average system utilization
Mateescu et al. (2011)	Based on a hybrid infrastructure for predictable execution of HPC workloads	Advance reservation	Ignored cost	VM allocation algorithm	Amazon cloud	Availability, performance
Huang et al. (2013)	Based on a decentralized scheduling algorithm without requiring detailed node information	Reduced average job slowdown	High complexity	Community-aware scheduling algorithm	Magate Simulation	Resource uptime, resource usage
Ghanbari and Othman (2012)	Based on priority of jobs for job scheduling	Considered multi-criteria	Lacks implementation on a real-world cloud platform	Priority based job scheduling	No experimental setup detail	Makespan algorithm
Li et al. (2012)	Based on resource allocation mechanism in cloud systems for preemptable tasks	Decreased energy consumption	Longer computational time	DCLS and DCMMS	Simulated environment	Average execution time, energy consumption
Ma et al. (2019)	Focused on resource allocation strategy	Reduced cost	Difficult to implement	Artificial bee colony	Simulation environment	Cost

Shenai et al. (2012) surveyed the scheduling algorithms in cloud computing. They did not consider different types of scheduling techniques and QoS parameters that can be used at the time of comparison for scheduling algorithms in cloud computing. Various resource allocation based scheduling algorithms are compared in Table 12. Huang et al. (2013) introduced a decentralized dynamic scheduling approach entitled the community-aware scheduling algorithm (CASA). In the proposed algorithm, the decision for job allocation is done by considering real-time responses of nodes. Rescheduling of the tasks facility is provided by CASA to handle unpredictable circumstances of resource allocations. They have used a set of heuristic methods to attain the target of optimized performance.

4.1.5. Miscellaneous techniques

Few resource scheduling algorithms are listened in miscellaneous techniques as these algorithms could not be listed in the above-discussed categories. Bi et al. (2017) focused on the issues of resource provisioning in a virtualized cloud data center. The goal of the research work is to reduce electricity prices and number of rejected requests between cloud consumers and providers. Chen et al. (2018) designed an unceRtainty-aware Online Scheduling Algorithm (ROSA) to schedule dynamic and multiple workflows with deadlines. Their main intent is to establish a good trade-off among cost, deviation, resource utilization, and fairness. Chen et al. (2017) developed a novel scheduling approach to schedule workflow tasks. Their main intent is to minimize both the makespans and monetary costs for executing workflows in clouds while improving VMs' resource utilization. Zhu et al. (2015) focused on the multi-objective optimization algorithm for scheduling workflows on IaaS. They have proposed a set of new genetic operators, the evaluation function and the population initialization scheme for task instance assignment problem by minimizing cost and time simultaneously. Alamer and Basudan (2020) designed secure placement of tasks for vehicles. They have tried to maintain the privacy of vehicles in Fog cloud computing environment. Shukla et al. (2020) addressed scheduling problems of computing systems. They have used the concept of fuzzy sets to solve the optimization problem while achieving energy efficiency. Burkimsher et al. (2013) surveyed scheduling metrics such as fairness, response time and resource utilization, etc. A Projected-Schedule Length Ratio (SLR) policy was designed using SLA metrics that helps in scheduling decisions. They have shown that SLR metric can be beneficial for mutually dependent workload as SLA metric helps to provide a critical path for the workload's execution.

Amiri and Mohammad-Khanli (2017) has presented state-of-art of application prediction models by considering different aspects for the execution of applications. A dynamic sampling model is designed for the estimation of the bandwidth for cloud scheduling problems. The base of the model is the data size and estimated capacity of the network in Yildirim et al. (2013). Their main intent is to provide optimization services to the application by considering data transfer parameters. Vasile et al. (2015) have explored the resource provisioning step that occurs before the scheduling. They have tried to accumulate the information of available resources and tasks before formal allocation of jobs on ingredient resources. In Sirbu et al. (2017), authors have used the stochastic process to provide information about resources for resource provisioning. In Thierens and Bosman (2013), authors have focused on the hierarchical problems to provide solution for complex system. They have used Linkage Tree Genetic Algorithm (LTGA) for hierarchical problems. Bosman et al. (2016) introduced Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA) family to solve permutation optimization problems by using the random keys encoding of permutations. Hsu and Yu (2015) proposed a new evolutionary algorithm, called DSMGA-II, to efficiently solve optimization problems via exploiting problem substructures. In the proposed algorithm, the pairwise linkage detection from DSMGA is used to design a new linkage model. Komarnicki et al. (2020) have proposed comparative mixing operator by extending the functionality of restricted mixing operator. The main intent of the proposed operator is to change linkage information, obtained by DSB based linkage model. Przewozniczek and Komarnicki (2020) focused on linkage learning techniques for local optimization. Saini et al. (2020) have discussed the new approached by using scalarization function to solve the optimization problems. Pan et al. (2021) used adaptation mechanism to design evolutionary algorithm to solve flow shop scheduling problem. Goldman and Punch

(2014) have focused on parameter-less population pyramid evolutionary technique to solve the optimization problem. They have used adaptive strategy to set feedback probability for each iteration. By using this method, parameter tuning is not required and very effective for overlapping problem in binary domain. In this section, all scheduling problem related to other applications domains are discussed.

4.2. Comparison of resource management systems

4.2.1. Eagle

The main motive of the eagle scheduler is to use the sticky batch probing at the time of scheduling. It works as a hybrid data center scheduler for data-parallel programs (Delgado et al., 2016). The main intent of the eagle is to do partitioning of the data center's node for the execution of short and long-duration jobs. Eagle job scheduler is very efficient for data center scheduling as data center scheduling is a very challenging task due to many reasons. The main issues in data center scheduling are handling the different types of workloads and the parallel nature of jobs and scaling of data centers. Eagle addresses all these issues for data center scheduling. Load balancing can be used with the help of eagle.

4.2.2. Hopper

Hopper (Ren et al., 2015) is a job scheduler that can be used to amalgamate the trade-offs associated with speculation into job scheduling decisions. It basically helps to dynamically allocate the capacity of extra slots by using the marginal value concept. Hopper considers many factors such as corporate data locality, fairness, DAGs of tasks at the time of job scheduling. It works for the centralized and decentralized prototype for both.

4.2.3. Kubernetes

The first version of Kubernetes was first released in June, 2014. Kubernetes is written in Go Language. Kubernetes is an open-source orchestrator developed by Google for automating container management and deployment (Anon, 2018a). Unlike Swarm, the basic deployable object here is a Pod which consists of one or several containers that run in a shared context. It is used to run and manage multiple docker containers with a REST API that allows users to declare how the various containers should scale and talk to each other. Kubernetes is an abstraction over an application which creates containers. With Kubernetes there is no direct contact with the actual servers that run the containers, we use the API for description such as what to run and how many copies of it. YAML-based deployment mode is used by Kubernetes. Many other features such as load balancing, auto-scaling, and secret management are provided by Kubernetes.

4.2.4. Tetris

Tetris (Grandl et al., 2015) is a cluster scheduler that aims to match multi-resource task requirements with resource availabilities. It considers both disk and network requirements of the application at the same time. Tetris provides many objectives such as fairness, minimizing job completion time, and minimizing cluster makespan for scheduling of multi-packing of the resources for tasks.

4.2.5. Fawkes

Fawkes (Ghit et al., 2014) is multi-cluster system. It is designed to deploy a new abstraction layer for mapreduce frameworks. The main aim of Fawkes is to do resource balancing dynamically. Provisioning strategies are used for scheduling of mapreduce jobs in multi clusters. All these strategies work on the basis of dynamic weights.

4.2.6. Omega

Omega (Schwarzkopf et al., 2013) is parallel scheduler architecture that uses the concept of parallelism, shared state, and lock-free optimistic concurrency control.

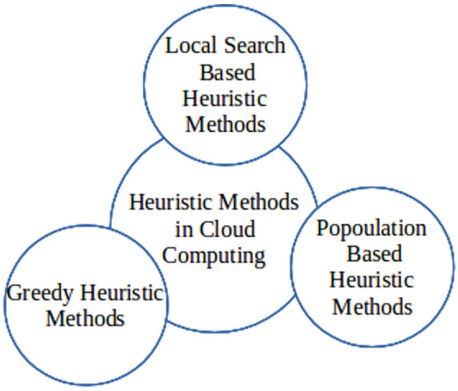


Fig. 3. Types of Heuristic Methods.

4.2.7. OurGrid instead of omega

OurGrid (Andrade et al., 2003; Cirne et al., 2006) is developed at Universida de Federal de Campina Grand, Brazil. OurGrid works as a centralized scheduler. The main vision of OurGrid is to bring together all labs for preparing a massive worldwide computing platform. It provides an isolated environment for application execution. Security constraints are very strict. OurGrid provides the facility of resource sharing in an equity fashion. OurGrid was designed to solve the issue of negotiation among users and resource providers. The main function of OurGrid is to be used for the execution of bag-of-tasks applications.

4.2.8. Sparrow

Sparrow (Ousterhout et al., 2013) is used to furnish the facility of fine-grained task scheduling. It acts as a complementary facility to resource managers. Sparrow does not initiate new processes for each task. It can be used for providing high throughput and lower latency based applications. The main intent of the sparrow is to do approximation whenever there is a trade-off among many features at the time of scheduling. Sparrow uses a randomized sampling approach for scheduling and bestows optimal performance using batch sampling and late binding techniques.

4.2.9. Yarn

Apache Hadoop YARN (Vavilapalli et al., 2013) acts as a resource management system. It is used to schedule the jobs in the open-source Hadoop distributed processing framework. The main liability of the YARN is to allocate the systems resources to different applications in the Hadoop cluster. It supports many QoS features such as scalability, higher efficiency, and fair resource usage. The two main components of resource manager are scheduler and applications manager. The main task of the scheduler is to allocate the resources to different applications as per the requirements as shown in Fig. 3. The capacity scheduler is used for predictable sharing of cluster resources by using hierarchical queues. job submissions, negotiating with the first container for executing the application is done by the application manager. Resource monitoring and container management would be performed by the node manager and the status of resources should be sent to the resource manager.

4.2.10. Mesos

Mesos (Hindman et al., 2011; Anon, 2018b) provides the API for resource management and scheduling in data centers. The main intent of the Mesos is to abstract CPU, memory, storage, and other compute resources away from machines. It enables the facility to build a distributed system fault-tolerant. Mesos, a thin resource sharing layer that helps to furnish fine-grained sharing by providing a common interface among different cluster frameworks. It helps the resource management system to achieve high utilization, respond quickly to workload changes, by maintaining the system's capability in terms of scalability and robustness.

Table 13
Comparison of different resource management systems.

RMS /Property	Architecture	Usage	Open source	Support	Applications	Programming framework
Eagle	Hybrid	Differentiates short and long jobs	EPFL IC IINFCOM LABOS, Switzerland	Spark	Different workloads and Parallel jobs	Python, Java, PHP, Python, Ruby
Hopper	Decentralized	Speculation-aware job scheduler	Microsoft Research	Spark	CPU intensive	Java, Php, .net
Kubernetes	Centralized	Fine-grained allocation in Docker containers	Cloud Native Computing Foundation and Google Borg	Generic Applications, Custom implementation for spark	CPU and Data intensive	Java, Python
Tetris	Centralized	Multi-resource bin-packing	Microsoft	Generic applications	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala,
Fawkes	Centralized	Dynamic resource balancing	TU Delft	Mapreduce frameworks	Data intensive	Python, Java, PHP, Python, Ruby
Omega	Decentralized	Shared state abstraction	University of Cambridge	Custom applications	Parallel applications	Java, Php, .net
OurGrid	Centralized	Equitable Resource Sharing	Universidade Federal de CampinaGrand, Brazil	Generic applications	Bag of Tasks	Java, Python
Sparrow	Decentralized	Randomized sampling approach	U.C. Berkeley AMPLab	Spark	CPU intensive	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala, Clojure, .Net
Yarn	Monolithic	Resource requests with containers	Hadoop	Spark	Data intensive	Python, Java, PHP, Python, Ruby, Clojour
Mesos	Two way protocol	Pessimistic resource offers	University of California, Berkeley	Spark	CPU and Data Intensive	Python, Perl, PHP, Rest, Ruby, .net, C#
HT Condor	Two way protocol	To execute computational workloads	University of Wisconsin–Madison	Custom applications	CPU intensive	Python, Perl, PHP

4.2.11. HTCondor

HTCondor (Thain et al., 2005) was designed at the University of Wisconsin. The main reason to design condor for distributed systems to provide the features such as consistency, availability, and performance for distributed applications. HTCondor is a specialized workload management system for compute-intensive jobs. It uses queueing approach for the scheduling of the jobs. There are many features available for resource management systems in condor like scheduling policy, priority scheme, resource monitoring. Jobs can be submitted in parallel forms and after the completion of the execution process, results will be sent back to users. The main use of the condor is for applications those demand high throughput computing. A comparison of different resource management system is shown in Table 13.

4.3. Scheduling in IaaS: State-of-art

In this section, scheduling in IaaS clouds has been compared with respect to their properties such as pricing plan, service level agreement, number of data centers, monitoring types, web APIs, and programming frameworks. We have considered seven cloud service providers for comparison of scheduling in IaaS clouds as shown in Table 14. By using IaaS the cloud consumers need not to invest in hardware upfront, so the development and deployment of applications became faster.

4.3.1. Amazon EC2

A Scalable computing facility is provided by Amazon Elastic Compute Cloud (Amazon EC2). In Amazon web service, cloud users can start virtual servers as per the requirement. There is no restriction on the number of virtual servers. Security and network configuration services are also provided by Amazon EC2. Whenever the demand for virtual servers increases, Amazon EC2 scales up or down as demand decreases (Anon, 2015).

For executing an application on elastic compute cloud, different types of VM instances are provided as per the requirement. Users can control costs also by stopping instances when not in use. EC2 scheduler is the main scheduler, used in AmazonEC2 for providing the facility of start and stop instances for completing the task. Resource capacity can also be controlled.

4.3.2. Microsoft Window Azure

Microsoft Window Azure is used for deploying, building, and managing applications and services. There are different types of services such as compute, storage, and networking that are provided by Azure. Different types of programming frameworks and tools are supported by Azure. The Azure scheduler is very robust. It provide the facility to create, update, delete the jobs. The Azure scheduler can be used for complex schedules such as daily maintenance, performing backups, and recurring application scenarios, etc. The main task of the Azure scheduler is to allow users to create jobs. Users can invoke the job inside or outside of the Azure environment by calling HTTP endpoints. The Azure scheduler switches to other data centers whenever the demand for resources is increased (Anon, 2014).

4.3.3. Rackspace

Rackspace, formerly known as Mosso, for providing the Infrastructure as a Service. The main task of the Rackspace is to provide the backup for data (Anon, 2016a). Scheduling is also considered an important aspect in the Rackspace. To create a scheduled task on the cloud site, five components are used. The round robin algorithm is mostly used to main the load balance among nodes. After creating each cron job, the task of selecting the best resource as per the job requirement is started.This algorithm helps in directing traffic in a particular manner to achieve the target of load balancing simultaneously. Encryption and compression facility is also provided by Rackspace.

Table 14
Comparison of different IaaS models.

Provider /Property	Pricing	SLA	Data Centers	Monitoring	Web APIs	Programming framework
Amazon EC2	Pay-as-you-go or Year, reserved, spot	99.95%	8	Good	Extensive	Python, Java, PHP, Python, Ruby
Microsoft window azure	Pay-as-you-go, semester, year	99.90%	8	Average	Good	Java, Php, .net
Rackspace	Pay-as-you-go	100%	6	Extensive	Good	Java, Python
HP Cloud	Pay as you go	99.95%	3	Poor	Average	Python, Perl, Java, PHP, Ruby, Node.js, Erlang, Scala, Clojure, .Net
Cloud sigma	Pay-as-you-go	100%	3	Good	Average	Python, Java, PHP, Python, Ruby, Clojour
Soft layer	Pay-as you go, monthly	99%	12	Extensive	Good	Python, Perl, PHP, Rest, Ruby, .net, C#
Go grid	Pay-as-you-go or Monthly, semester, year	100%	5	Poor	None	Python, Perl, PHP

4.3.4. HP cloud

HP cloud solutions help to build and operate own private cloud with HP cloud system. Their main aim is to deliver a highly secure environment for enterprise workloads and leverage the HP public cloud for open, enterprise-class storage, compute, and networking solutions.

In the HP cloud, the main task of the scheduler is to identify the node for the task as per the requirement. HPE Helion OpenStack 4.0, freezer scheduler help in managing all operation related to scheduling. Open stack nova concept, compute host facility is used by freezer scheduler. Nova concept helps in the process of identifying logical separation within the cloud on the basis of physical isolation. The Nova scheduler provides the facility of filtering and weighting to make scheduling decisions (Anon, 2016b).

4.3.5. Cloud sigma

The main intent of the cloud sigma is to maintain the balance between public and private cloud service providers. The virtualization of Cloud sigma is based on KVM and it also supports para-virtualization too. It offers full-spectrum solution to cloud users to balance their workloads. Cloud sigma can be used for critical applications such as disaster recovery management, early forest fore detection, to attain low latency targets by combining private technology. Resource management system of cloud sigma helps to decrease the cost of an application execution on the public cloud. For the execution of the application, cloud sigma provides the solution of networking as a service and secure data center facility too. Cloud sigma is considered as the best choice for hybrid hosting due to the patch up with private technology (Anon, 2016c).

4.3.6. Soft layer

SoftLayer provides cloud infrastructure as a service solution, proposed by IBM Company (Anon, 2016d). There are many advantages associated with soft layers as it is an easy-to-use platform, having robust API for accessing all services. It also supports network within network topology structure. In softlayer, users can submit 2600 jobs per minute. Softlayer scheduler is efficient in handling all the jobs dynamically. Scheduler having capacity planning activity to predict the workload in advance. Due to this feature, horizontal scaling can be done. In the updated version of cloud sigma, vertical scaling is possible by adding one dynamic workload broker who handles the task of adding resources as per the resource utilization parameters.

4.3.7. GoGrid

GoGrid helps in cloud hosting on both windows and Linux cloud servers. It provides all services on infrastructure as a service space. The main intent of GoGrid is to build a secure hybrid infrastructure for web hosting as per the user's requirement. Load balancing is also supported by GoGrid to attain the optimal balance among nodes (Anon, 2016e). In GoGrid, the job can be submitted on dedicated servers, directly managed by the user. After the job submission, dedicated servers are not shared with other entities. These dedicated servers reside on the private network.

All important parameters of IaaS clouds such as pricing plan, service level agreement, number of data centers, monitoring types, web APIs, and programming frameworks have been discussed. Table 8 comparison of scheduling in IaaS clouds.

- Pricing Plan: This attribute represents the estimated cost in US$ for 1 CPU and 1 GB Ram. It does not include data transfer costs. There are different types of plans such as monthly plans, hourly plans, yearly and discounts for members. Users can select plans as per the requirement for application's execution. Pay-as-you go model is considered the best pricing plan.
- Service Level Agreement: SLA means a mutual agreement between cloud consumers and cloud service providers. For comparison, we have considered CPU uptime SLA offered by different cloud service providers.
- Number of Data centers: It states the number of data centers offered as a choice when users want to deploy cloud server.
- Monitoring Type: In this field, monitoring of resources is considered after the starting application's execution process. Few cloud service providers take the monitoring type of service from third parties to provide the monitoring solutions.
- Web APIs : This field describes the company offers APIs to interact with the servers or not to cloud users.
- Programming Framework: In the programming framework, we have listed down the language which cloud users can use for the execution of an application on different service providers as shown in Table 14.

4.4. Heuristics and meta-heuristics methods for cloud scheduling

The process of mapping jobs to ingredient resources in cloud scheduling is called an NP-complete problem (Braun et al., 2001). Heuristic methods are used to solve the NP-complete problems. There are many issues related to cloud scheduling such as heterogeneity of the resources, dynamic and autonomous nature of cloud resources, which can be addressed by using heuristic methods. Different heuristic methods are studied and compared in Braun et al. (2001). Fig. 3 shows the categorization of existing heuristic methods in cloud computing.

4.4.1. Greedy heuristic approaches

Greedy algorithms are called intuitive heuristics. The reason to called intuitive heuristic is that greedy choices are selected to achieve the main target (Merz and Freisleben, 2002). In greedy heuristic, feasible solutions are constructed from scratch by selecting appropriate choices in each step. To solve optimization problems, there is a need to add elements in the partial solution that helps to provide the highest gain.

Opportunistic Load Balancing: In opportunistic Load Balancing (OLB), the main intent of the algorithm is to place each job, in order of arrival, on the next available machine. The expected execution time of the task for that particular machine is computed (Armstrong et al., 1998). The main target of OLB is to keep all machines busy and it does not consider the execution time which results in poor makespan.

Minimum Execution Time: In minimum execution time heuristic methods, tasks are assigned to machines as per arrival order. While assigned task, the best expected execution time is calculated (Braun et al., 2001). The main aim of the minimum execution time method is to assign the task best machine for the execution of the application. There are high chances of load imbalance between processors as it does not consider the load of the resources.

Minimum Completion Time: The main motive of Minimum Completion Time (MCT) is to reap the advantages of both OLB and MET. In the minimum completion time algorithm, each task is assigned to the machines in arbitrary order. While assigning the task, the minimum expected completion time for each task is emphasized (Armstrong et al., 1998).

Min–min: In the min–min heuristic method, all tasks with the known set of minimum completion time is considered. The main base of the min method is the minimum completion time. After checking all un-mapped tasks, the task with the overall minimum completion is considered to assign the machine (Armstrong et al., 1998; Freund et al., 1998). In each mapping decision, all un-mapped tasks will be considered but the minimum completion task for the one task is focused at a time.

Max–min: In max–min heuristic methods, all un-mapped task will be considered and the task with the overall maximum completion time is selected for the allocation of the machine. The main intent of max–min methods is to reduce the overhead of tasks having high execution time (Armstrong et al., 1998; Freund et al., 1998). In max–min, load balancing factor is considers for the better utilization of the resources. For example, one application having one task with a high execution time and the rest task having a shorter execution time. The max–min method will allocate the higher execution to the best available machine so that all other shorter running tasks can run simultaneously. But this is not the case with min–min heuristic methods.

Duplex: Duplex heuristic method is the merger of max–min and min–min heuristic methods. Both min–min and max–min algorithms are checked for a better solution. Then the task will be assigned to a particular machine (Braun et al., 2001).

4.4.2. Meta-heuristics

Meta-heuristics algorithms having an iterative process that instructs the operations of subordinate heuristic methods to get the better solutions. These methods are used to assist in the decision process of many applications such as scientific, engineering, business, and economic within a defined time frame (Stefan, 2001). There many advantages associated with meta-heuristic methods as these methods are flexible to solve many real-time problems, robust, and uses global optimizer. The main drawback of the meta-heuristic method is that result can vary for the same problem when different methods are applied. There is no theoretical approach so that optimality is not guaranteed. Metaheuristics required extensive knowledge of heuristic and problem domain to get the good solution.

Local Search based Meta-Heuristic Methods: Local search heuristic methods explore the solution space by selecting an initial solution. To construct the path for solution space, each solution is explored during the searching process (Xhafa and Abraham, 2010). The main intent of the local search heuristic method is to find the feasible solution quickly by searching neighborhood solution space and local search heuristic approaches (Cappanera and Trubian, 2005). These methods are used to solve industrial-level problems.

Tabu Search: Tabu Search (TS) is a meta-heuristic method, proposed by Gover in 1986. TS methods explore solution space by avoiding the trap of local minima (Glover, 1989). There are many advantages associated with tabu search methods as it is very simple in comparison to other meta-heuristic methods such as genetic algorithm, simulated annealing, and particle swarm optimization, etc. It is used to solve the combinatorial optimization problems. There is one drawback with this method that it has stepped into the trap of local optima while shifting from one local optima to another due to low global search capability.

Hill Climbing: Hill climbing heuristic method is a local search-based method. It is very simple to implement in comparison to other heuristic methods. This heuristic method is implemented in those situations where the current path along with the successor node can be extended. It is a useful method as there are no more than one maxima and minima (Xhafa and Abraham, 2010). The main drawback of this method is that the solution is better in comparison to the neighborhood area but not in the another solution space area.

Simulated Annealing: Simulated Annealing (SA) heuristic method is an iterative technique in which one solution is considered. Kirkpatrick et al. designed a simulated annealing method in 1983 (Gelatt et al., 1983). In this method, system is melted at a high temperature for optimization and then the temperature will be slow at the various levels until the system freezes (Theys et al., 2001). Simulated annealing methods use probabilistic approach to get the optimal solution. The initial implementation of SA is modified to get the refined solution (Braun et al., 2001). There are many benefits to use this approach as it is easy to code, guaranteed to converge in asymptotic time. SA can be applied to multi-objective optimization problems due to the robust nature of the method. The main difficulty of SA is to select multiple parameters and time consumption is more. It is not easy to define the cooling temperature too.

Population based Meta-Heuristic approaches: Population-based heuristic methods are used to solve scheduling problems in a distributed computing environment such as Grid and cloud computing (Xhafa and Abraham, 2010).

Genetic Algorithms: Genetic Algorithm (GA) was designed for solving optimization problems, by Holland (1975). It is a stochastic optimization algorithm, used to provide an optimal solution (Theys et al., 2001). Genetic algorithm can be used to solve issues in performance tuning and classification applications. In genetic algorithm, there are different operations such as genetic inheritance, crossover, mutation, and reproduction (Hsu, 2004). In the first step, a set of parameters are defined in search space to represent the genes. A set of genes is considered as a chromosome. A set of chromosome is defined as population. In the second step, the fitness value of each chromosome will be evaluated. The fitness function is designed as per the problem requirement (Abraham et al., 2000). On the basis of fitness values, genes will be selected for the reproduction. In the third step, the reproduction step will be performed by taking two genes to pass their genes to the next generation.

Crossover is the most important step of genetic algorithm. After selecting two parents, the crossover will be performed in the fourth step. In the fifth step, the mutation will be done to ensure that there is no point with zero probability in the search space. It avoids premature convergence. Genetic algorithm terminates if the population has converged achieved global optimum solution (Kokilavani and Amalarethinam, 2010; Braun et al., 2001; Judy and Ramadoss, 2012). In a cloud computing environment, genetic-based heuristic methods are used to solve the scheduling problem (Sfrent and Pop, 2015). There are many advantages of GA as it does not require analytical knowledge and derivatives

during implementation. GA can easily apply for solving large-scale combinatorial optimization problems. There is a major drawback of premature convergence issue. Genetic algorithm only provides local optimum solutions to the problem in a short span of time.

Memetic Algorithm: Memetic Algorithm (MA) is considered as an extended version of genetic algorithm. The advantages of both local search methods and genetic algorithm methods are combined in MA. These algorithms are applied in local search methods to get the refined solution. As the base of the memetic algorithm is a genetic algorithm, the population is varied to avoid premature convergence. It is used to solve real-world problems like university timetables, in the prediction process of protein structures, and the design of space-craft trajectories (Hart et al., 2004). The main use of MA is to solve the non-linear continuous multi-objective combinatorial optimization problems.

Ant Colony Optimization: In 1992, Marco Dorigo has proposed Ant Colony Optimization (ACO) (Colorni et al., 1991). The main strength of the ACO algorithm is that ants keep real power in their colony brain as it is found in brain-like structures. On their way, ants release a molecule of pheromone to alert the other ants to go in one direction (Merloti, 2004). Due to non-deterministic algorithm, ACO relies on other sub heuristic algorithm to find out the optimal solution (Dorigo and Gambardella, 1997). There are numerous advantages of ACO (Dorigo et al., 1996): ACO is versatile algorithm that can be applied to solve job shop scheduling problem and quadratic assignment problem. After doing the few changes, ACO helps to solve static and dynamic combinatorial optimization problems. Due to good convergence criteria, ACO can help to solve the discrete problem too. ACO meta-heuristic method can easily adapt as per real-time for applications. There is one major drawback of ACO as there is no centralized control for solving traveling salesman problem. Theoretical analysis of ACO is also difficult.

Particle Swarm Optimization: Particle Swarm Optimization (PSO) was introduced in 1995 by Kennedy (2011). In PSO, there is no need for the explicit knowledge of gradient to solve the problem. PSO helps in stimulation, i.e., the process of a swarm bird praying. PSO is applied to solve numerical optimization problems due to the ability of global searching. A group of particle is randomly generated. In PSO, the first step is to represent each particle as a possible solution. In the second step, each particle's fitness value will be evaluated as per the designed objective function. In the third step, if the particle's fitness is the best from global solution (gbest) then gbest will be updated. Then particle will be evaluated with the current best solution(pbest). In the fourth step, if the particle is better then the current best solution is updated. Each particle moves in the group to find the best solution. In the fifth step, the velocity vector will be used to update the position of each particle. There are many advantages associated with PSO as it is a robust optimization method due to many factors as there are no crossover and mutation operators. It is easy to implement and having global search ability. PSO cannot be applied for few applications due to the slow convergence rate.

Bacterial Foraging Optimization: Bacterial Foraging Optimization (BFO) algorithm was designed by Passino (2002). BFO is based on the foraging behavior of Escherichia coli bacteria. In the foraging concept, the main intent of the Escherichia coli bacteria is to find out the nutrients in such a manner so that energy intake per unit time (E/T) is maximized. In the foraging process, each bacteria move region to region in the search of the food to get maximum energy. There are three basic steps performed in BFO: Chemotaxis, Reproduction, and Elimination-dispersal event. In the chemotaxis step, the behavior of E. coli bacteria is stimulated by doing operation swimming and tumbling. E. coli bacteria release attractants to provide the alert message to other bacteria to swarm together. In the second step, the fitness of each bacteria will be evaluated. The week bacteria will die with time and healthy bacteria will produce two new bacteria so that swarm size will be constant. In the last step, elimination will occur due to sudden change such as an increase in temperature etc. Elimination and dispersal steps can react in both ways as it can be useful in the scenario when bacteria is trapped in the useless region or it can be harmful when bacteria was already in a healthy region for food (Dasgupta et al., 2009). There are a lot of advantages of BFO algorithm as it is more adaptive in comparison to ACO, PSO. There is no issue of premature convergence as it can be applied to real-world problems such as projector scheduling, resource scheduling, etc. BFO is easy to implement as there is no complex operator for the computation.

A comparison of heuristic methods is shown in Table 15. After analysis, it can be concluded that BFO is a more appropriate method for resource scheduling in cloud computing in comparison to other heuristic methods. BFO provides the more optimal solution.

4.4.3. Hybrid-heuristics

Hybrid methods are designed to exploit the main advantages of meta-heuristic methods. Genetic algorithm is combined with local search methods such as tabu search, hill climbing, and simulated annealing, etc to get the optimal results. Hybrid heuristic provides better convergence and more efficient in comparison to other algorithms. Due to simplicity nature of hybrid heuristic methods, it can be easily implemented in shared memory parallel architectures (Talbi, 2002). The main drawback of the hybrid heuristic method is that it is not an easy to implement and time-consuming method.

4.4.4. Hyper-heuristics

Hyper-heuristic selects heuristic from the set of low-level heuristic to solve the combinatorial optimization problems. It is considered as a high-level methodology that provides a set of low-level heuristics as per the problem requirements (Burke et al., 2013). There is no need for extra knowledge of low-level heuristic at each decision point to solve the problem.

The main advantage of the hyper-heuristic algorithm is that it adapts to the problem environment. Chakhlevitch and Cowling (2008) has defined few problems while implementing hyper-heuristic algorithm. Few hyper-heuristic methods are verified on benchmark problems such as traveling salesman problem, project planning problem, etc. It also requires few parameters to tune at the advance stage for the better selection of low-level heuristic to solve real-world problems.

After the thorough analysis of heuristic methods, it has been concluded that local search methods cannot be applied to solve all large-scale problems to get the optimal solution in the reasonable time frame. So, meta-heuristic methods can be sued to solve such problems but it requires an ample amount of knowledge of the problem and expensive to solve the problem too. In comparison to meta-heuristic, hyper-heuristic methods can be used to solve combinatorial optimization problems (Burke et al., 2013). The key difference between meta-heuristic and hyper-heuristic is that meta-heuristics start directly working on the problem to find out the solution but hyper-heuristic method generates low-level heuristic to solve the problem as per the problem specifications. Hyper-heuristic is considered a more viable methodology in-comparison to other heuristic methods such as hybrid heuristic, meta-heuristics. For distributed environments like grid and cloud computing, hyper-heuristic methods are considered appropriate for solving scheduling optimization problems.

5. Open issues and challenges

Recently, the promising performance of cloud resource management system has allured attention in the cloud computing environment. The state-of-the-art in the considered field was presented in the previous sections and it is found that there are many issues and challenges that need to be addressed. In future studies, there is a need to work on the unexplored issues. In this section, open issues and prospective research directions have also been discussed. After studying the existing work, few important key issues are listed as follows:

Table 15
Comparison of different heuristic approaches.

Heuristics/ Features	Parameters	Convergence	Premature convergence	Services	Local/Global search	Optimization problems
Tabu search	Less parameters	Guaranteed convergence	Prevent premature convergence	conceptual, Simpler, easy to implement, no special memory requirement	Low global search	Combinatorial optimization
Hill climbing	Less functions	No guaranteed	Prevent premature convergence	Simpler and straight forward	Local search	Simple Optimization Problem
Simulated annealing	Less Functions	Converge In asymptotic time	Premature convergence	Easy to code, robust heuristic	Local search	Combinatorial optimization Problems
Genetic algorithm	More functions	No guaranteed	premature Convergence	No need analytical knowledge, easy to run and implement	Global search capability	Multi objective optimization
Memetic algorithm	More functions	Guaranteed convergence	Less chance of premature Convergence	Flexible	Global search	Complex objective functions, non-linear multi objective Combinatorial optimization Problems
Ant colony optimization	Less functions	Guaranteed convergence	avoid the premature Convergence	Versatile, robust	Global search	Static and dynamic Combinatorial optimization Problems
Particle swarm optimization	No function like genetic algorithm	Slow convergence rate	Less chance of premature Convergence	Robust	Global search	Stochastic optimization
Bacterial foraging optimization	No function like genetic algorithm	Better convergence rate	Avoid premature Convergence	Flexible & Robust	Global search	Real-world optimization

Load Balancing: Efficient distribution of the jobs/tasks across geographically distributed cloud computing environment can reduce the costs by dynamically reallocating the workload to a place where the computing resources and access to computing resources are cheaper. Load balancing is also one of the major part of cloud resource management and scheduling. VM migrations and server consolidation techniques can be used at the time of load balancing. If the workload is distributed evenly then it will enhance the performance of the system. There would be less chance of node failure and user will get the result of application's execution on time at a very less price as cloud offers as you use you pay. There is a need to work to maintain the balance between data-locality and load-balancing to maximize throughput and minimize delay simultaneously.

Resource Utilization: Resource utilization is also one of the most important QoS parameters for cloud scheduling. A cloud scheduling algorithm should be well equipped to enhance the resource utilization in an efficient and effective manner. It is a required to design resource utilization aware techniques to reduce cost and increase the speed of the application's execution. In a cloud computing environment, the cloud resources and the user requests can change dynamically. Therefore, a scheduling approach should be smart enough to make real-time responses to a changing environment. A multi-objective nature is inherent in cloud resource scheduling, as the objectives of cloud providers, cloud users, and other stakeholders can be independent. For example, a cloud service provider may try to minimize the cost of the application's execution via a resource utilization-aware scheduling algorithm. Even among cloud users themselves, different users or the same user at different times may have different QoS requirements, such as minimum computational costs, faster execution of applications, and so on. Therefore, a multi-objective based resource scheduling problem could become more and more significant in the future of cloud computing.

Edge Computing and Fog Computing: Due to the increased edge and fog computing computational power available, significant activities can also be conducted on edge devices. Many IoT devices are connected and do computation a lot. Thus, scheduling must balance between activities that can take place on the edge (fog) or needs to be conducted in the cloud.

Scheduling Challenges arising from use of Containers: Container used virtualization technology to provide the abstraction at OS level. Containers are joined in a virtual network. The challenge here is to assure that containers between users do not create security or violate privacy issues. Also, the access to potentially elevated system privileges may cause other issues. Therefore, systems such as singularity offer users an isolated use of containers within traditional HPC queuing systems to mitigate that issue. Such challenges must be integrated into a scheduling strategy when adding containerized cloud resources.

Challenge in Serverless Computing: To handle the growing traffic, the cost of infrastructure is the big issue that needs to be addressed in serverless computing paradigm. There is a need to design scheduling algorithm to meet the demand of the infrastructure efficiently.

Risk Analysis: Risk management on a daily basis is a thing of the past as there are higher volatiles. To maintain competitive advantages in real-life scenarios like in banking, health sector; there is a need to evaluate their models continuously, including the performance of the production models. Another aspect to analyze risk in a cloud computing environment helps cloud consumers assessing their risk before putting their critical data in a security sensitive cloud. Work needs to be done in establishing the relationship between risk matrices (trust, SLA violation ratio, availability, and elasticity) and SLA to provide effective risk management to the users.

Security: Security is a crucial concern in cloud computing. Data security is an open issue in the cloud computing environment as there is no access to security systems of data centers to the cloud service providers. Cloud service providers have to depend on the infrastructure service provider to take access to the security system. Even for a virtual private cloud, the cloud provider can only identify the security setting distantly, without knowing whether it is completely implemented or not. It is dangerous to form trust procedures at each architectural layer of the cloud. Initially, the hardware layer must be reliable using hardware reliable platform segment. Furthermore, the virtualization platform needs to be confidential using secure VM observers. VM migration should only be permitted if both sender and receiver servers are confidential. There are many issues related to security aspects such as privacy-preserving computations on outsourced, encrypted data in the cloud computing environment. Whenever two-party computation occurs, thus allowing a wide applicability that can range from private genome processing to cloud computing. In order to improve the security of the confidentiality information, there is a need to lower the high communication complexity between the cloud and the proxy server.

Energy Efficiency: In cloud data centers, energy consumption is one of the major problems. Some research projects have already started to investigate advanced energy-aware resource management systems to build a strong foundation of prior works in cloud computing. The virtual machines communicate with one another in a network of different topologies. If the allocation of resources is not done in an optimized way, then many migrations of processes will occur. Data transfer would be increased because CPUs are logically hosted on distant physical servers. In this case, the communication could become a bottle-neck as it involves switches, access points, and routers that also consume more power. So it will also give results in form of delaying packets because packets requisite to travel across the network. To eliminate the data transfer delays, costs and reduce power consumption, setting an appropriate communication pattern among CPUs is important. Therefore, more work is needed to place CPUs on the same or different cloud to reduce energy consumption.

It has been assessed that the price of powering and refrigeration accounts for 53% of the entire operational spending of data centers. In 2006, data centers in the US consumed more than 1.5% of the total energy produced in that year, and the proportion is estimated to grow 18% yearly. Consequently, cloud service providers started to look into the area of how can they decrease energy consumption via considering government rules and environmental standards. Energy-aware task scheduling and server consolidation techniques are two good procedures that help to decrease energy consumption by releasing ideal systems.

Big Data: Cloud computing has not been flourished as a successful business model that has been widely adopted by the enterprises to store their big data assets. As structured and unstructured data is increasing at a very high speed, to handle a large amount of complex data, cloud computing can provide a solution Data as a Service (DAAS) in the future. Analysis of big data is a very complex task so there is a need to design a tool for analyzing the big data. As it demands many steps such as cleaning, structuring, understanding, choosing proper methods, and analyzing the results. There are mainly five critical issues related to the growth of big data in cloud computing such as 1. Bottlenecks in Data Transfer, 2. Performance Unpredictability, 3. Scalable Storage, 4. Occurrence of Bugs in Large Distribution Systems and 5. Quick Scaling On Demand. Everything from the ability to run analytically at a scale in a virtual environment to ensuring information processing and analytics authenticity are the issues that need solutions and have to be fixed. Designing scalable, elastic, and autonomic multi-tenant database systems is another important challenge that must also be addressed. In addition, ensuring the security and privacy of the data outsourced to the cloud is also an important problem for ensuring the success of data management systems in the cloud.

Analytics as a Service: Cloud-based data analytics has been growing rapidly in an effort to reap all the benefits of the clouds. There are two main research challenges: (1) big data analytics and (2) fast data analytics. There is a need to provide analytics as a service by reaping the advantages of cloud computing. Streamline analytics is an on-demand, cloud-based, analytics and reporting service, which transforms data into easy-to-use, actionable dashboards, benchmarks, and metrics-driven analytics. Traditionally, hospital reports and analytics were generated on a one-off basis, often requiring resource-heavy and inefficient processes. Frequently limited to selected departments and personnel, the data was often out of date and difficult to interpret and use. Streamline analytics techniques are required for quick response. It requires a secure, integrated framework for aggregating financial, operational, and clinical data from disparate systems. There is a need to investigate and develop technology to provide Analytics-as-a-Service (AaaS). Several challenges are involved in order to build a platform to provide AaaS, which include SLA definitions, QoS monitoring techniques, pricing, analysis and management of unstructured data, and business models. Analytics-as-a-Service issues need to be addressed through three main categories: descriptive, predictive, and prescriptive. There is no information in streamline analytics.

Other Issues: Some of the current challenges are suggested as follows: (i) Real-Time Location Intelligence and Recommendation Engines; (ii) Cloud Computing and Service-Oriented Thinking; (iii) Component-based Service Orientation Fosters in Cloud Computing for Reusability, Substitutability, Extensibility, Scalability, Customizability, Reliability, Low Cost of Ownership and Economy of Scale.

6. Conclusions

In this paper, we have presented the basic concept of cloud scheduling, their types, and the need of scheduling in the cloud computing environment. We surveyed scheduling problems along with their solutions in the cloud computing. After analyzing scheduling problems, it can be concluded that cloud scheduling problem is more complex in comparison to other distributed platforms such as grid computing and cluster computing, etc. It also reveals that the hyper-heuristic methods are more appropriate to solving the scheduling problem in the cloud environment. The detailed study of heuristic and meta-heuristic methods is also done along with their importance for the design of efficient scheduling algorithms. This paper gives a clear image for choosing the best heuristic method for addressing cloud resource scheduling problems. Open issues and challenges are also discussed in detail. This will help researchers to frame the research problems in the field of cloud resource management systems.

CRediT authorship contribution statement

Rajni Aron: Conceptualization, Methodology, Taxonomy preparation, Writing – original draft. **Ajith Abraham:** Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was supported by the Analytical Center for Government of the Russian Federation under Grant 70-2021-00143 dd. 01.11.2021 and Grant IGK000000D730321P5Q0002.

References

Abraham, A., Buyya, R., Nath, B., 2000. Nature's heuristics for scheduling jobs on computational grids. In: The 8th IEEE International Conference on Advanced Computing and Communications. ADCOM 2000, pp. 45–52.

Abrishami, S., Naghibzadeh, M., Epema, D.H., 2013. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener. Comput. Syst. 29 (1), 158–169.

Adhikari, M., Amgoth, T., Srirama, S.N., 2020. Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach. Appl. Soft Comput. 106411.

Afoulki, Z., Bousquet, A., Rouzaud-Cornabas, J., 2011. A security-aware scheduler for virtual machines on iaas clouds. Report 2011.

Akbar, M., Irohara, T., 2020. Metaheuristics for the multi-task simultaneous supervision dual resource-constrained scheduling problem. Eng. Appl. Artif. Intell. 96, 104004.

Al-khateeb, A., Abdullah, R., et al., 2009. Job type approach for deciding job scheduling in grid computing systems. J. Comput. Sci. 5 (10), 745–750.

Alamer, A., Basudan, S., 2020. An efficient truthfulness privacy-preserving tendering framework for vehicular fog computing. Eng. Appl. Artif. Intell. 91, 103583.

Amiri, M., Mohammad-Khanli, L., 2017. Survey on prediction models of applications for resources provisioning in cloud. J. Netw. Comput. Appl. 82, 93–113.

Andrade, N., Cirne, W., Brasileiro, F., Roisenberg, P., 2003. OurGrid: An approach to easily assemble grids with equitable resource sharing. In: Workshop on Job Scheduling Strategies for Parallel Processing. Springer, pp. 61–86.

Anon, 2014. Microsoft Azure. URL https://azure.microsoft.com/en-in/services/scheduler/.

Anon, 2015. Amazon EC2. URL https://aws.amazon.com/ec2/.

Anon, 2016a. Rackspace. URL https://www.rackspace.com/en-in/why-rackspace.

Anon, 2016b. HP cloud. URL https://docs.hpcloud.com/hos-4.x/helion/operations/compute/creating_aggregates.htm.

Anon, 2016c. Cloud sigma. URL https://www.cloudsigma.com/features/.

Anon, 2016d. Softlayer. URL http://www.softlayer.com/about-softlayer.

Anon, 2016e. GoGrid. URL https://www.datapipe.com/gogrid/.

Anon, 2018a. Kubernates. URL https://github.com/kubernetes/kubernetes.

Anon, 2018b. Mesos. URL http://mesos.apache.org/getting-started/.

Arabnejad, V., Bubendorfer, K., 2015. Cost effective and deadline constrained scientific workflow scheduling for commercial clouds. In: Network Computing and Applications (NCA), 2015 IEEE 14th International Symposium on. IEEE, pp. 106–113.

Arabnejad, V., Bubendorfer, K., Ng, B., 2016. A budget-aware algorithm for scheduling scientific workflows in cloud. In: High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on. IEEE, pp. 1188–1195.

Arabnejad, V., Bubendorfer, K., Ng, B., 2017. Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources. Future Gener. Comput. Syst. 75, 348–364.

Ari, I., Muhtaroglu, N., 2013. Design and implementation of a cloud computing service for finite element analysis. Adv. Eng. Softw. 60, 122–135.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al., 2010. A view of cloud computing. Commun. ACM 53 (4), 50–58.

Armstrong, R., Hensgen, D., Kidd, T., 1998. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh. IEEE, pp. 79–87.

Arunarani, A., Manjula, D., Sugumaran, V., 2019. Task scheduling techniques in cloud computing: A literature survey. Future Gener. Comput. Syst. 91, 407–415.

Attiya, I., Abd Elaziz, M., Abualigah, L., Nguyen, T.N., Abd El-Latif, A.A., 2022. An improved hybrid swarm intelligence for scheduling IoT application tasks in the cloud. IEEE Trans. Ind. Inf..

Beloglazov, A., Abawajy, J., Buyya, R., 2012. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Gener. Comput. Syst. 28 (5), 755–768.

Beloglazov, A., Buyya, R., 2010. Energy efficient resource management in virtualized cloud data centers. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE Computer Society, pp. 826–831.

Bessis, N., Sotiriadis, S., Pop, F., Cristea, V., 2013. Using a novel message-exchanging optimization (MEO) model to reduce energy consumption in distributed systems. Simul. Model. Pract. Theory 39, 104–120.

Bi, J., Yuan, H., Tan, W., Li, B.H., 2016. TRS: Temporal request scheduling with bounded delay assurance in a green cloud data center. Inform. Sci. 360, 57–72.

Bi, J., Yuan, H., Tan, W., Zhou, M., Fan, Y., Zhang, J., Li, J., 2017. Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center. IEEE Trans. Autom. Sci. Eng. 14 (2), 1172–1184.

Bi, J., Yuan, H., Tie, M., Tan, W., 2015. SLA-based optimisation of virtualised resource for multi-tier web applications in cloud data centres. Enterp. Inf. Syst. 9 (7), 743–767.

Bilogrevic, I., Jadliwala, M., Kumar, P., Walia, S.S., Hubaux, J.-P., Aad, I., Niemi, V., 2011. Meetings through the cloud: Privacy-preserving scheduling on mobile devices. J. Syst. Softw. 84 (11), 1910–1927.

Biran, O., Corradi, A., Fanelli, M., Foschini, L., Nus, A., Raz, D., Silvera, E., 2012. A stable network-aware vm placement for cloud systems. In: Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on. IEEE, pp. 498–506.

Bosman, P.A., Luong, N.H., Thierens, D., 2016. Expanding from discrete cartesian to permutation gene-pool optimal mixing evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 637–644.

Bousselmi, K., Brahmi, Z., Gammoudi, M.M., 2016. Qos-aware scheduling of workflows in cloud computing environments. In: Advanced Information Networking and Applications (AINA), 2016 IEEE 30th International Conference on. IEEE, pp. 737–745.

Braun, T.D., Siegel, H.J., Beck, N., Bölöni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D., et al., 2001. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J. Parallel Distrib. Comput. 61 (6), 810–837.

Breitgand, D., Epstein, A., 2012. Improving consolidation of virtual machines with risk-aware bandwidth oversubscription in compute clouds. In: INFOCOM, 2012 Proceedings IEEE. IEEE, pp. 2861–2865.

Bui, D.-M., Yoon, Y., Huh, E.-N., Jun, S., Lee, S., 2016. Energy efficiency for cloud computing system based on predictive optimization. J. Parallel Distrib. Comput..

Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. J. Oper. Res. Soc. 64 (12), 1695–1724.

Burkimsher, A., Bate, I., Indrusiak, L.S., 2013. A survey of scheduling metrics and an improved ordering policy for list schedulers operating on workloads with dependencies and a wide variation in execution times. Future Gener. Comput. Syst. 29 (8), 2009–2025.

Buyya, R., Vecchiola, C., Selvi, S.T., 2013. Mastering Cloud Computing: Foundations and Applications Programming. Elsevier, Amsterdam,The Netherlands.

Calheiros, R.N., Buyya, R., 2014. Energy-efficient scheduling of urgent bag-of-tasks applications in clouds through DVFS. In: Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on. IEEE, pp. 342–349.

Cappanera, P., Trubian, M., 2005. A local-search-based heuristic for the demand-constrained multidimensional knapsack problem. INFORMS J. Comput. 17 (1), 82–98.

Chakhlevitch, K., Cowling, P., 2008. Hyperheuristics: Recent developments. In: Adaptive and Multilevel Metaheuristics. Springer, pp. 3–29.

Chejerla, B.K., Madria, S.K., 2017. Qos guaranteeing robust scheduling in attack resilient cloud integrated cyber physical system. Future Gener. Comput. Syst. 75, 145–157.

Chen, H., Zhu, X., Liu, G., Pedrycz, W., 2018. Uncertainty-aware online scheduling for real-time workflows in cloud service environment. IEEE Trans. Serv. Comput. 14 (4), 1167–1178.

Chen, H., Zhu, X., Qiu, D., Liu, L., Du, Z., 2017. Scheduling for workflows with security-sensitive intermediate data by selective tasks duplication in clouds. IEEE Trans. Parallel Distrib. Syst. 28 (9), 2674–2688.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L.B., Andrade, A., Novaes, R., Mowbray, M., 2006. Labs of the world, unite!!! J. Grid Comput. 4 (3), 225–246.

Colorni, A., Dorigo, M., Maniezzo, V., et al., 1991. Distributed optimization by ant colonies. In: Proceedings of the First European Conference on Artificial Life, Vol. 142. Paris, France, pp. 134–142.

Dabbagh, M., Hamdaoui, B., Guizani, M., Rayes, A., 2015. Energy-efficient resource allocation and provisioning framework for cloud data centers. IEEE Trans. Netw. Serv. Manag. 12 (3), 377–391.

Dasgupta, S., Das, S., Abraham, A., Biswas, A., 2009. Adaptive computational chemotaxis in bacterial foraging optimization: an analysis. Evol. Comput. IEEE Trans. on 13 (4), 919–941.

Delgado, P., Didona, D., Dinu, F., Zwaenepoel, W., 2016. Job-aware scheduling in eagle: Divide and stick to your probes. In: Proceedings of the Seventh ACM Symposium on Cloud Computing. ACM, pp. 497–509.

Devi, K.L., Valli, S., 2021. Multi-objective heuristics algorithm for dynamic resource scheduling in the cloud computing environment. J. Supercomput. 1–29.

Ding, Y., Qin, X., Liu, L., Wang, T., 2015. Energy efficient scheduling of virtual machines in cloud with deadline constraint. Future Gener. Comput. Syst. 50, 62–74.

Dong, F., Akl, S.G., 2006. Scheduling Algorithms for Grid Computing State of the Art and Open Problems. Technical report.

Dorigo, M., Gambardella, L.M., 1997. Ant colonies for the travelling salesman problem. BioSystems 43 (2), 73–81.

Dorigo, M., Maniezzo, V., Colorni, A., 1996. Ant system: Optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. B 26 (1), 29–41.
Duan, H., Chen, C., Min, G., Wu, Y., 2017. Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems. Future Gener. Comput. Syst. 74, 142–150.
Duan, Y., Lu, Z., Zhou, Z., Sun, X., Wu, J., 2019. Data privacy protection for edge computing of smart city in a DIKW architecture. Eng. Appl. Artif. Intell. 81, 323–335.
Erdil, D.C., 2013. Autonomic cloud resource sharing for intercloud federations. Future Gener. Comput. Syst. 29 (7), 1700–1708.
Foster, I., Zhao, Y., Raicu, I., Lu, S., 2008. Cloud computing and grid computing 360-degree compared. In: Grid Computing Environments Workshop, 2008. GCE'08, Ieee, pp. 1–10.
Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., 2009. Above the clouds: A berkeley view of cloud computing. Dept. Electrical Eng. and Comput. Sci. Univ. Calif. Berkeley, Rep. UCB/EECS 28 (13), 2009.
Freund, R.F., Gherrity, M., Ambrosius, S., Campbell, M., Halderman, M., Hensgen, D., Keith, E., Kidd, T., Kussow, M., Lima, J.D., et al., 1998. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet. In: Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh. IEEE, pp. 184–199.
Frîncu, M.E., 2014. Scheduling highly available applications on cloud environments. Future Gener. Comput. Syst. 32, 138–153.
Fu, Y., Hou, Y., Wang, Z., Wu, X., Gao, K., Wang, L., 2021. Distributed scheduling problems in intelligent manufacturing systems. Tsinghua Sci. Technol. 26 (5), 625–645.
Gai, K., Qiu, M., Zhao, H., Tao, L., Zong, Z., 2016. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. J. Netw. Comput. Appl. 59, 46–54.
Garg, S.K., Yeo, C.S., Anandasivam, A., Buyya, R., 2011. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. J. Parallel Distrib. Comput. 71 (6), 732–749.
Gąsior, J., Seredyński, F., 2016. Metaheuristic approaches to multiobjective job scheduling in cloud computing systems. In: Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on. IEEE, pp. 222–229.
Geelan, J., et al., 2009. Twenty one experts define cloud computing. Cloud Comput. J. 4, 1–5.
Gelatt, C., Vecchi, M., et al., 1983. Optimization by simulated annealing. Science 220 (4598), 671–680.
Ghahramani, M.H., Zhou, M., Hon, C.T., 2017. Toward cloud computing QoS architecture: Analysis of cloud systems and cloud services. IEEE/CAA J. Autom. Sin. 4 (1), 6–18.
Ghanbari, S., Othman, M., 2012. A priority based job scheduling algorithm in cloud computing. Procedia Eng. 50, 778–785.
Ghit, B., Yigitbasi, N., Iosup, A., Epema, D., 2014. Balanced resource allocations across multiple dynamic MapReduce clusters. In: ACM SIGMETRICS Performance Evaluation Review, Vol. 42, no. 1. ACM, pp. 329–341.
Glover, F., 1989. Tabu search-Part I. ORSA J. Comput. 1 (3), 190–206.
Goldman, B.W., Punch, W.F., 2014. Parameter-less population pyramid. In: Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation. pp. 785–792.
Goyal, A., Dadizadeh, S., 2009. A Survey on Cloud Computing. University of British Columbia Technical Report for CS, 508, pp. 55–58.
Grandl, R., Ananthanarayanan, G., Kandula, S., Rao, S., Akella, A., 2015. Multi-resource packing for cluster schedulers. ACM SIGCOMM Comput. Commun. Rev. 44 (4), 455–466.
Gutierrez-Garcia, J.O., Sim, K.M., 2013. A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling. Future Gener. Comput. Syst. 29 (7), 1682–1699.
Hart, W.E., Krasnogor, N., Smith, J.E., 2004. Recent Advances in Memetic Algorithms, Vol. 166. Springer Science & Business Media.
Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A.D., Katz, R.H., Shenker, S., Stoica, I., 2011. Mesos: A platform for fine-grained resource sharing in the data center. In: NSDI, Vol. 11, no. 2011. pp. 22–22.
Holland, J.H., 1975. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. U Michigan Press.
Houssein, E.H., Gad, A.G., Wazery, Y.M., Suganthan, P.N., 2021. Task scheduling in cloud computing based on meta-heuristics: Review, taxonomy, open challenges, and future trends. Swarm Evol. Comput. 100841.
Hsu, W.H., 2004. Genetic Algorithms. Tech. Rep. 66506–2302, Department of Computing and Information Sciences, Kansas State University, 234 Nichols Hall, Manhattan, KS, USA.
Hsu, S.-H., Yu, T.-L., 2015. Optimization by pairwise linkage detection, incremental linkage set, and restricted/back mixing: DSMGA-II. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 519–526.
Hu, J., Gu, J., Sun, G., Zhao, T., 2010. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on. IEEE, pp. 89–96.
Hu, Z., Li, D., 2021. Improved heuristic job scheduling method to enhance throughput for big data analytics. Tsinghua Sci. Technol. 27 (2), 344–357.
Huang, Y., Bessis, N., Norrington, P., Kuonen, P., Hirsbrunner, B., 2013. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm. Future Gener. Comput. Syst. 29 (1), 402–415.
Irwin, D., Shenoy, P., Cecchet, E., Zink, M., 2010. Resource management in data-intensive clouds: opportunities and challenges. In: Local and Metropolitan Area Networks (LANMAN), 2010 17th IEEE Workshop on. IEEE, pp. 1–6.
Jennings, B., Stadler, R., 2015. Resource management in clouds: Survey and research challenges. J. Netw. Syst. Manage. 23 (3), 567–619.
Jing, W., Liu, Y., Shao, H., 2015. Reliability-aware DAG scheduling with primary-backup in cloud computing. Int. J. Comput. Appl. Technol. 52 (1), 86–93.
Joseph, J., 2009. Cloud Computing-Patterns for high availability, scalability, and computing power with windows azure. MSDN Mag. 60.
Judy, M., Ramadoss, B., 2012. An enhanced solution to the protein folding problem using a hybrid genetic algorithm with G-bit improvement strategy. Int. J. Model. Optim. 2 (3), 356.
Kashyap, R., Vidyarthi, D.P., 2014. Security-aware real-time scheduling for hypervisors. In: Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on. IEEE, pp. 1520–1527.
Keller, G., Tighe, M., Lutfiyya, H., Bauer, M., 2014. A hierarchical, topology-aware approach to dynamic data centre management. In: Network Operations and Management Symposium. NOMS, 2014 IEEE, IEEE, pp. 1–7.
Kennedy, J., 2011. Particle swarm optimization. In: Encyclopedia of Machine Learning. Springer, pp. 760–766.
Khojasteh Toussi, G., Naghibzadeh, M., 2021. A divide and conquer approach to deadline constrained cost-optimization workflow scheduling for the cloud. Cluster Comput. 24 (3), 1711–1733.
Kim, N., Cho, J., Seo, E., 2014. Energy-credit scheduler: An energy-aware virtual machine scheduler for cloud systems. Future Gener. Comput. Syst. 32, 128–137.
Kokilavani, T., Amalarethinam, D.D.G., 2010. Applying non-traditional optimization techniques to task scheduling in grid computing–An overview. Int. J. Res. Rev. Comput. Sci. (IJRRCS) 1, 33–38.
Komarnicki, M.M., Przewozniczek, M.W., Durda, T.M., 2020. Comparative mixing for DSMGA-II. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference. pp. 708–716.
Kondikoppa, P., Chiu, C.-H., Cui, C., Xue, L., Park, S.-J., 2012. Network-aware scheduling of mapreduce framework ondistributed clusters over high speed networks. In: Proceedings of the 2012 Workshop on Cloud Services, Federation, and the 8th Open Cirrus Summit. ACM, pp. 39–44.
Konjaang, J.K., Xu, L., 2021. Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review. J. Netw. Syst. Manage. 29 (2), 1–57.
Kousiouris, G., Cucinotta, T., Varvarigou, T., 2011. The effects of scheduling, workload type and consolidation scenarios on virtual machine performance and their prediction through optimized artificial neural networks. J. Syst. Softw. 84 (8), 1270–1291.
Kumar, M., Sharma, S.C., Goel, A., Singh, S.P., 2019. A comprehensive survey for scheduling techniques in cloud computing. J. Netw. Comput. Appl. 143, 1–33.
Latiff, M.S.A., Madni, S.H.H., Abdullahi, M., et al., 2016. Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm. Neural Comput. Appl. 1–15.
LD, D.B., Krishna, P.V., 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13 (5), 2292–2303.
Lee, G., Katz, R.H., 2011. Heterogeneity-aware resource allocation and scheduling in the cloud. In: HotCloud.
Lee, Y.C., Wang, C., Zomaya, A.Y., Zhou, B.B., 2012. Profit-driven scheduling for cloud services with data access awareness. J. Parallel Distrib. Comput. 72 (4), 591–602.
Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z., 2012. Online optimization for scheduling preemptable tasks on IaaS cloud systems. J. Parallel Distrib. Comput. 72 (5), 666–677.
Li, L., Shao, W., Zhou, X., 2021. A flexible scheduling algorithm for the 5th-generation networks. Intell. Converged Netw. 2 (2), 101–107.
Li, W., Tordsson, J., Elmroth, E., 2011. Modeling for dynamic cloud scheduling via migration of virtual machines. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on. IEEE, pp. 163–171.
Li, H., Zhu, H., Ren, G., Wang, H., Zhang, H., Chen, L., 2016. Energy-aware scheduling of workflow in cloud center with deadline constraint. In: Computational Intelligence and Security (CIS), 2016 12th International Conference on. IEEE, pp. 415–418.
Liu, C., Zhang, X., Yang, C., Chen, J., 2013. CCBKE session key negotiation for fast and secure scheduling of scientific applications in cloud computing. Future Gener. Comput. Syst. 29 (5), 1300–1308.
Lucas-Simarro, J.L., Moreno-Vozmediano, R., Montero, R.S., Llorente, I.M., 2013. Scheduling strategies for optimal service deployment across multiple clouds. Future Gener. Comput. Syst. 29 (6), 1431–1441.
Lucrezia, F., Marchetto, G., Risso, F., Vercellone, V., 2015. Introducing network-aware scheduling capabilities in openstack. In: Network Softwarization (NetSoft), 2015 1st IEEE Conference on. IEEE, pp. 1–5.
Ma, K., Liu, X., Li, G., Hu, S., Yang, J., Guan, X., 2019. Resource allocation for smart grid communication based on a multi-swarm artificial bee colony algorithm with cooperative learning. Eng. Appl. Artif. Intell. 81, 29–36.

Malik, S., Huet, F., Caromel, D., 2012. Reliability aware scheduling in cloud computing. In: Internet Technology and Secured Transactions, 2012 International Conference for. IEEE, pp. 194–200.

Mangla, N., et al., 2021. Resource scheduling on basis of cost-effectiveness in cloud computing environment. In: Mobile Radio Communications and 5G Networks. Springer, pp. 429–442.

Mateescu, G., Gentzsch, W., Ribbens, C.J., 2011. Hybrid computing where HPC meets grid and cloud computing. Future Gener. Comput. Syst. 27 (5), 440–453.

Mateos, C., Pacini, E., Garino, C.G., 2013. An ACO-inspired algorithm for minimizing weighted flowtime in cloud-based parameter sweep experiments. Adv. Eng. Softw. 56, 38–50.

Mell, P., Grance, T., 2009. The NIST definition of cloud computing. Natl. Inst. Stand. Technol. 53 (6), 50.

Merloti, P.E., 2004. Optimization algorithms inspired by biological ants and swarm behavior. San Diego State University, Citeseer.

Merz, P., Freisleben, B., 2002. Greedy and local search heuristics for unconstrained binary quadratic programming. J. Heuristics 8 (2), 197–213.

Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y.C., Talbi, E.-G., Zomaya, A.Y., Tuyttens, D., 2011. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. 71 (11), 1497–1508.

Nathani, A., Chaudhary, S., Somani, G., 2012. Policy based resource allocation in IaaS cloud. Future Gener. Comput. Syst. 28 (1), 94–103.

Ousterhout, K., Wendell, P., Zaharia, M., Stoica, I., 2013. Sparrow: distributed, low latency scheduling. In: Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles. ACM, pp. 69–84.

Owusu, F., Pattinson, C., 2012. The current state of understanding of the energy efficiency of cloud computing. In: Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on. IEEE, pp. 1948–1953.

Pan, Z.-X., Wang, L., Chen, J.-F., Wu, Y.-T., 2021. A novel evolutionary algorithm with adaptation mechanism for fuzzy permutation flow-shop scheduling. In: 2021 IEEE Congress on Evolutionary Computation. CEC, IEEE, pp. 367–374.

Pandey, S., Wu, L., Guru, S.M., Buyya, R., 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, pp. 400–407.

Passino, K.M., 2002. Biomimicry of bacterial foraging for distributed optimization and control. Control Syst. IEEE 22 (3), 52–67.

Priya, V., Kumar, C.S., Kannan, R., 2019. Resource scheduling algorithm with load balancing for cloud service provisioning. Appl. Soft Comput. 76, 416–424.

Przewozniczek, M.W., Komarnicki, M.M., 2020. Empirical linkage learning. IEEE Trans. Evol. Comput. 24 (6), 1097–1111.

Quarati, A., Clematis, A., Galizia, A., D Agostino, D., 2013. Hybrid clouds brokering: Business opportunities, QoS and energy-saving issues. Simul. Model. Pract. Theory 39, 121–134.

Rajni, Chana, I., 2010. Resource provisioning and scheduling in grids: issues, challenges and future directions. In: Computer and Communication Technology (ICCCT), 2010 International Conference on. IEEE, pp. 306–310.

Rajni, Chana, I., 2013. Bacterial foraging based hyper-heuristic for resource scheduling in grid computing. Future Gener. Comput. Syst. 29 (3), 751–762.

Rampersaud, S., Grosu, D., 2016. Sharing-aware online virtual machine packing in heterogeneous resource clouds. IEEE Trans. Parallel Distrib. Syst. 28 (7), 2046–2059.

Ren, X., Ananthanarayanan, G., Wierman, A., Yu, M., 2015. Hopper: Decentralized speculation-aware cluster scheduling at scale. In: ACM SIGCOMM Computer Communication Review, Vol. 45, no. 4. ACM, pp. 379–392.

Rittinghouse, J.W., Ransome, J.F., 2009. Cloud Computing: Implementation, Management, and Security. CRC Press.

Saini, B.S., Hakanen, J., Miettinen, K., 2020. A new paradigm in interactive evolutionary multiobjective optimization. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 243–256.

Sandhu, A.K., 2021. Big data with cloud computing: Discussions and challenges. Big Data Min. Anal. 5 (1), 32–40.

Schwarzkopf, M., Konwinski, A., Abd-El-Malek, M., Wilkes, J., 2013. Omega: Flexible, scalable schedulers for large compute clusters. In: Proceedings of the 8th ACM European Conference on Computer Systems. ACM, pp. 351–364.

Sfrent, A., Pop, F., 2015. Asymptotic scheduling for many task computing in big data platforms. Inform. Sci. 319, 71–91.

Shamsi, J., Khojaye, M.A., Qasmi, M.A., 2013. Data-intensive cloud computing: Requirements, expectations, challenges, and solutions. J. Grid Comput. 11 (2), 281–310.

Shenai, S., et al., 2012. Survey on scheduling issues in cloud computing. Procedia Eng. 38, 2881–2888.

Shetty, S., Yuchi, X., Song, M., 2016. Security-aware virtual machine placement in cloud data center. In: Moving Target Defense for Distributed Systems. Springer, pp. 13–24.

Shukla, A.K., Nath, R., Muhuri, P.K., Lohani, Q.D., 2020. Energy efficient multi-objective scheduling of tasks with interval type-2 fuzzy timing constraints in an Industry 4.0 ecosystem. Eng. Appl. Artif. Intell. 87, 103257.

Singh, R.M., Awasthi, L.K., Sikka, G., 2022. Towards metaheuristic scheduling techniques in cloud and fog: An extensive taxonomic review. ACM Comput. Surv. 55 (3), 1–43.

Sirbu, A., Pop, C., Şerbănescu, C., Pop, F., 2017. Predicting provisioning and booting times in a metal-as-a-service system. Future Gener. Comput. Syst. 72, 180–192.

Stefan, V., 2001. Meta-heuristics:The state of the art. In: Nareyek, A. (Ed.), Local Search for Planning and Scheduling. In: LNAI 2148, Springer-Verlag, Berlin Heidelberg, pp. 1–23.

Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., Wang, J., 2013. Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput. 39 (4), 177–188.

Sun, G., Liao, D., Zhao, D., Xu, Z., Yu, H., 2015. Live migration for multiple correlated virtual machines in cloud-based data centers. IEEE Trans. Serv. Comput. 11 (2), 279–291.

Talbi, E.-G., 2002. A taxonomy of hybrid metaheuristics. J. Heuristics 8 (5), 541–564.

Tang, X., Tan, W., 2016. Energy-efficient reliability-aware scheduling algorithm on heterogeneous systems. Sci. Program. 2016, 14.

Thain, D., Tannenbaum, T., Livny, M., 2005. Distributed computing in practice: The condor experience. Concurr. Comput.: Pract. Exper. 17 (2–4), 323–356.

Theys, M.D., Braun, T.D., Siegal, H., Maciejewski, A.A., Kwok, Y., 2001. Mapping tasks onto distributed heterogeneous computing systems using a genetic algorithm approach. In: Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences. John Wiley & Sons, New York, NY, pp. 135–178.

Thierens, D., Bosman, P.A., 2013. Hierarchical problem solving with the linkage tree genetic algorithm. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 877–884.

Thomas, A., Krishnalal, G., Raj, V.J., 2015. Credit based scheduling algorithm in cloud computing environment. Procedia Comput. Sci. 46, 913–920.

Tighe, M., Bauer, M., 2014. Integrating cloud application autoscaling with dynamic vm allocation. In: Network Operations and Management Symposium. NOMS, 2014 IEEE, IEEE, pp. 1–9.

Tighe, M., Keller, G., Bauer, M., Lutfiyya, H., 2013. A distributed approach to dynamic VM management. In: Network and Service Management (CNSM), 2013 9th International Conference on. IEEE, pp. 166–170.

Torabzadeh, E., Zandieh, M., 2010. Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flowshop. Adv. Eng. Softw. 41 (10), 1238–1243.

Tordsson, J., Montero, R.S., Moreno-Vozmediano, R., Llorente, I.M., 2012. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. Future Gener. Comput. Syst. 28 (2), 358–367.

Tsai, C.-W., Rodrigues, J.J., 2014. Metaheuristic scheduling for cloud: A survey. Syst. J. IEEE 8 (1), 279–291.

Van den Bossche, R., Vanmechelen, K., Broeckhove, J., 2013. Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. Future Gener. Comput. Syst. 29 (4), 973–985.

Van Do, T., Rotter, C., 2012. Comparison of scheduling schemes for on-demand IaaS requests. J. Syst. Softw. 85 (6), 1400–1408.

Vasile, M.-A., Pop, F., Tutueanu, R.-I., Cristea, V., Kołodziej, J., 2015. Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing. Future Gener. Comput. Syst. 51, 61–71.

Vavilapalli, V.K., Murthy, A.C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al., 2013. Apache hadoop yarn: Yet another resource negotiator. In: Proceedings of the 4th Annual Symposium on Cloud Computing. ACM, p. 5.

Vivekanandan, K., et al., 2011. A study on scheduling in grid environment. In: International Journal on Computer Science and Engineering. IJCSE, Citeseer.

Wang, W., Zeng, G., Tang, D., Yao, J., 2012. Cloud-DLS: Dynamic trusted scheduling for cloud computing. Expert Syst. Appl. 39 (3), 2321–2329.

Wu, L., Garg, S.K., Buyya, R., 2012. SLA-based admission control for a software-as-a-service provider in cloud computing environments. J. Comput. System Sci. 78 (5), 1280–1299.

Xhafa, F., Abraham, A., 2010. Computational models and heuristic methods for grid scheduling problems. Future Gener. Comput. Syst. 26 (4), 608–621.

Xu, B., Zhao, C., Hu, E., Hu, B., 2011. Job scheduling algorithm based on berger model in cloud environment. Adv. Eng. Softw. 42 (7), 419–425.

Yildirim, E., Kim, J., Kosar, T., 2013. Modeling throughput sampling size for a cloud-hosted data scheduling and optimization service. Future Gener. Comput. Syst. 29 (7), 1795–1807.

Yu, R., Xue, G., Zhang, X., Li, D., 2017. Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers. In: IEEE INFOCOM.

Yuan, H., Bi, J., Tan, W., Li, B.H., 2016. CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers. IEEE Trans. Autom. Sci. Eng. 13 (2), 976–985.

Yuan, H., Bi, J., Tan, W., Li, B.H., 2017a. Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds. IEEE Trans. Autom. Sci. Eng. 14 (1), 337–348.

Yuan, H., Bi, J., Tan, W., Zhou, M., Li, B.H., Li, J., 2017b. TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds. IEEE Trans. Cybern. 47 (11), 3658–3668.

Yuan, H., Bi, J., Zhou, M., Ammari, A.C., 2017c. Time-aware multi-application task scheduling with guaranteed delay constraints in green data center. IEEE Trans. Autom. Sci. Eng. 15 (3), 1138–1151.

Yuan, H., Bi, J., Zhou, M., Sedraoui, K., 2018. WARM: Workload-aware multi-application task scheduling for revenue maximization in SDN-based cloud data center. IEEE Access 6, 645–657.

Zeng, L., Veeravalli, B., Li, X., 2015. SABA: A security-aware and budget-aware workflow scheduling strategy in clouds. J. Parallel Distrib. Comput. 75, 141–151.

Zhan, Z.-H., Liu, X.-F., Gong, Y.-J., Zhang, J., Chung, H.S.-H., Li, Y., 2015. Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. Surv. 47 (4), 63.

Zhang, P., Zhou, M., 2018. Dynamic cloud task scheduling based on a two-stage strategy. IEEE Trans. Autom. Sci. Eng. 15 (2), 772–783.

Zhu, Z., Zhang, G., Li, M., Liu, X., 2015. Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. 27 (5), 1344–1357.

Zhu, W., Zhuang, Y., Zhang, L., 2017. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. Future Gener. Comput. Syst. 69, 66–74.

Zuo, L., Shu, L., Dong, S., Zhu, C., Hara, T., 2015. A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access 3, 2687–2699.