Swarm and Evolutionary Computation 62 (2021) 100841

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Survey Paper

Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends

Essam H. Houssein [a,*], Ahmed G. Gad [b], Yaser M. Wazery [a], Ponnuthurai Nagaratnam Suganthan [c]

[a] Faculty of Computers and Information, Minia University, Minia, Egypt
[b] Faculty of Computers and Information, Kafrelsheikh University, Kafrelsheikh, Egypt
[c] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

ARTICLE INFO

Keywords:
Cloud computing
Task scheduling
Meta-heuristics
Optimization
Quality of Service (QoS)
Simulation tools
Open challenges
Future trends
Systematic review

ABSTRACT

Cloud computing is a recently looming-evoked paradigm, the aim of which is to provide on-demand, pay-as-you-go, internet-based access to shared computing resources (hardware and software) in a metered, self-service, dynamically scalable fashion. A related hot topic at the moment is task scheduling, which is well known for delivering critical cloud service performance. However, the dilemmas of resources being underutilized (underloaded) and overutilized (overloaded) may arise as a result of improper scheduling, which in turn leads to either wastage of cloud resources or degradation in service performance, respectively. Thus, the idea of incorporating meta-heuristic algorithms into task scheduling emerged in order to efficiently distribute complex and diverse incoming tasks (cloudlets) across available limited resources, within a reasonable time. Meta-heuristic techniques have proven very capable of solving scheduling problems, which is fulfilled herein from a cloud perspective by first providing a brief on traditional and heuristic scheduling methods before diving deeply into the most popular meta-heuristics for cloud task scheduling followed by a detailed systematic review featuring a novel taxonomy of those techniques, along with their advantages and limitations. More specifically, in this study, the basic concepts of cloud task scheduling are addressed smoothly, as well as diverse swarm, evolutionary, physical, emerging, and hybrid meta-heuristic scheduling techniques are categorized as per the nature of the scheduling problem (i.e., single- or multi-objective), the primary objective of scheduling, task-resource mapping scheme, and scheduling constraint. Armed with these methods, some of the most recent relevant literature are surveyed, and insights into the identification of existing challenges are presented, along with a trail to potential solutions. Furthermore, guidelines to future research directions drawn from recently emerging trends are outlined, which should definitely contribute to assisting current researchers and practitioners as well as pave the way for newbies excited about cloud task scheduling to pursue their own glory in the field.

1. Introduction

In cloud computing, to optimize one or more scheduling objectives (e.g., makespan, computational cost, monetary cost, reliability, availability, resource utilization, response time, energy consumption, etc.), a scheduler (broker) [1,2] is developed in order to determine potential solutions for allotting a set of available limited resources to incoming tasks/applications. In general, Johnson's study [3] is believed to be the historical basis of modern scheduling approaches. Nowadays, many different applications are built based on the scheduling concept, such as power system control, scheduling of multimedia data objects on the Internet, as well as manufacturing printed circuit boards [2]. Distributed computing systems have undergone several developments since their very inception in 1980 as the allocation of limited resources on these systems to tasks submitted by Internet users is one of the major applications of modern scheduling. The emergence of cluster systems is one of these recent changes, in which a variety of standalone computers are combined so that they have the potential to work jointly as one system [4]. Since only local resources are used by cluster systems, grid systems have been developed later to overcome this shortcoming by bringing together all heterogeneous resources available in geographically distributed areas [5]. Transitioning to cloud systems [6] is a recent change in which the strengths of both cluster and grid systems are leveraged well.

Most problems of scheduling are either NP-hard or NP-complete [7] in which, due to the large solution space, a long time is required to implement an optimal or sub-optimal solution within a minimal

* Corresponding author.
E-mail addresses: essam.halim@mu.edu.eg (E.H. Houssein), ahmed.gad@fci.kfs.edu.eg (A.G. Gad), yaser.wazery@minia.edu.eg (Y.M. Wazery), epnsugan@ntu.edu.sg (P.N. Suganthan).

https://doi.org/10.1016/j.swevo.2021.100841
Received 1 May 2020; Received in revised form 7 October 2020; Accepted 9 January 2021
Available online 21 January 2021

period of time. Thus, unfortunately, there is no existing polynomial time-scheduling algorithm that can be applied to optimize the scheduling of limited resources in modern computing systems [8]. For illustrating the dilemma we are facing, a simple example was presented by Taillard [9], in which around 0.02% of the candidate solutions consume between 1 and 1.01 the total time needed to obtain the optimum solution. This example shows the extreme difficulty of finding the optimum solution to complex problems. Consequently, pursuing a good algorithm has been triggered by most researchers to find out a fast yet efficient solution to such type of scheduling problems. Two basic types of scheduling techniques are static and dynamic techniques. But cloud environments are dynamic in nature; therefore, it is essential that more dynamic algorithms are incorporated in order to achieve breathtaking results in the cloud scheduling area. Contrariwise, static algorithms are only used when the workloads have a small variation. Thus, solving the task scheduling problem [10,11] using deterministic methods is not practical. Meta-heuristic algorithms, being non-deterministic methods, have been offered to significantly solve this problem in a polynomial period of time.

Multiple benefits from dynamic task scheduling techniques and virtualization technology can be leveraged by cloud service users and providers. Effective resource (task) scheduling not only reduces resource consumption (increases the resource utilization ratio) but also executes incoming tasks in minimum time (minimizes the makespan). Scheduling of tasks has become of great importance due to the scarcity of resources in cloud that might result from the ongoing increase in workloads at cloud datacentres. Hence, more research is needed in the nascent field of cloud task scheduling in order to push for: a more efficient mapping of incoming tasks to available resources, as well as improving the Quality of Service (QoS) parameters. The main aim of a novel scheduling technique is to evaluate the optimum set of resources available to execute an incoming task such that a scheduling algorithm (scheduler) can then be applied to optimize such diverse QOS parameters as cost, makespan, scalability, reliability, task rejection ratio, resource utilization, energy consumption, etc., and fulfill constraints, like deadline, budget, etc., aiming at avoiding the problem of load imbalance (underutilization and overutiliztion), while maintaining the Service Level Agreement (SLA).

1.1. Motivation behind the research

Generally, current computer systems have two common scheduling methods including exhaustive algorithms and Deterministic Algorithms (DAs [12]). DAs are faster for scheduling problems, so they are much preferable in practical terms than both traditional (exhaustive) and heuristic algorithms. However, DAs have two main disadvantages: i) they are developed not for all the data distributions, and ii) large-scale scheduling problems cannot be tackled by most DAs. Unlike DAs and exhaustive algorithms, meta-heuristic algorithms (or approximation algorithms) find optimum solutions in a reasonable time by employing iterative strategies. Meta-heuristic scheduling algorithms give better scheduling results than traditional and heuristic ones as evidenced by numerous research results [13–15]. However, their primary interest is not cloud computing. Despite the successful application of many available scheduling methods to diverse computing environments, like grid and clustering computing [16], as well as the potential direct use of some of these methods in cloud environments, they are not explicitly developed for cloud. Thus, to the public, meta-heuristics may, at first glance, may seem inappropriate as a scheduling choice for cloud tasks. Motivated by this misbelief, this study not only introduces a systematical description of scheduling methods in the cloud environment from a meta-heuristics perspective, but also establishes a link between traditional/heuristic scheduling techniques and meta-heuristic ones so as to enable cloud researchers who remain passionate about traditional/heuristic scheduling to easily shift to scheduling based on meta-heuristics. Here, it should be pointed out that the goal of addressing both traditional and heuristic algorithms before debating modern heuristics (meta-heuristics) is to facilitate the distinction between them.

We have been also motivated by impulses stemming from peer surveys in the previous literature. Task scheduling is a key aspect of cloud computing, which strives to maximize Virtual Machines' (VMs') utilization while reducing the datacentres' operational cost, which in turn reveals significant improvements in the QoS parameters and the overall performance. A large number of user requests can be properly handled and scheduled to suitable VMs with the help of a proper task scheduling technique, which contributes to fulfilling the requirements of cloud users and service providers more efficiently. We closely examined many meta-heuristic scheduling approaches in the literature, and it has been found that the application of those algorithms in cloud computing has been triggered in a few prominent surveys [17–23]. The approach proposed in each survey is described as per Table 1. For example, some meta-heuristic techniques for load balancing were briefly discussed and compared against each other based on one key parameter, makespan, by Garg and Kumar [17]; however, other general as well as survey parameters, like taxonomy, open issues, recent trends, etc., are not undertaken in this work. On the other hand, only the QoS parameters and the existing challenges of scheduling techniques in cloud environment were put forward by Singh et al. [24]. In a similar work, existing meta-heuristic scheduling algorithms were briefly summarized in a survey by Nandhakumar and Ranjithprabhu [21] and Singh et al. [24] based on the QoS parameters. Nevertheless, only a limited number of parameters (e.g., year-wise analysis, state-of-the-art, QoS parameters, etc.) are considered in all the aforementioned papers. Table 1 contains a few untouched parameters.

Moreover, in [18] and [20], the survey techniques were improved better: taxonomy, graphical representation, and more QoS parameters are considered; however, all parameters are not simultaneously analyzed in the existing surveys as shown in Table 1. Hence, the field of task scheduling in the cloud is scarce for research and scrutinization. Therefore, a comprehensive survey on task scheduling using meta-heuristics is urgently needed, to be in line with the ongoing growing research in the field. As per Fig. 1, meta-heuristics have rapidly evolved over the last ten years, thereby forming a strong trend in the cloud task scheduling paradigm. In order to catch up with this growth, to the best of our knowledge, a comprehensive, systematic, taxonomic review of meta-heuristic scheduling approaches in the cloud is presented in this study. In addition, potential future trends are identified based on the existing research challenges faced by the selected approaches.

1.2. Our contributions

Although meta-heuristic scheduling techniques have a significant effect on the cloud service; as far as we know, important techniques and backgrounds of the field are yet to be comprehensively and systematically reviewed. Accordingly, this study aims to examine and analyze existing cloud scheduling approaches with respect to the meta-heuristic algorithms and differentiate between the picked techniques in order to finally give some pivotal issues that can be tackled in the future, along with some recommended emerging trends in the cloud scheduling area. In order to make the rationale clearer, Fig. 2 is created to succinctly describe the structure of this study. Concisely, the present survey could help move the area forward by assisting young researchers and practitioners to apply existing scheduling algorithms or approaches or to devise new ones. This paper involves main focal points that can be summed up as follows:

1. Diverse existing well-known traditional, heuristic, and meta-heuristic algorithms for task scheduling in cloud are examined and analyzed, giving however a special emphasis to meta-heuristics.
2. A novel classification scheme (taxonomy) as well as rigorous review of up-to-date meta-heuristic scheduling techniques in cloud computing are presented.

Table 1
Differentiating between present study and existing studies.

Publication	QoS-based comparative analysis	Year-wise comparison	State-of-the-art	Taxonomy	Graphical representations	Open issues	Future trends	Comparison of simulation tools
Garg and Kumar [17]	✗	✗	✗	✗	✗	✗	✗	✗
Kalra and Singh [18]	✗	✗	√	√	✗	√	✗	✗
Kaur and Chhabra [19]	✗	✗	√	✗	✗	√	✗	✗
Tsai and Rodrigues [20]	✗	✗	√	√	✗	√	✗	✗
Nandhakumar and Ranjithprabhu [21]	√	✗	✗	√	✗	✗	✗	√
Kapur [22]	✗	✗	√	√	✗	✗	✗	✗
Shishira et al. [23]	✗	✗	√	√	✗	√	✗	✗
Singh et al. [24]	√	✗	√	√	✗	√	✗	✗
Present study	√	√	√	√	√	√	√	√

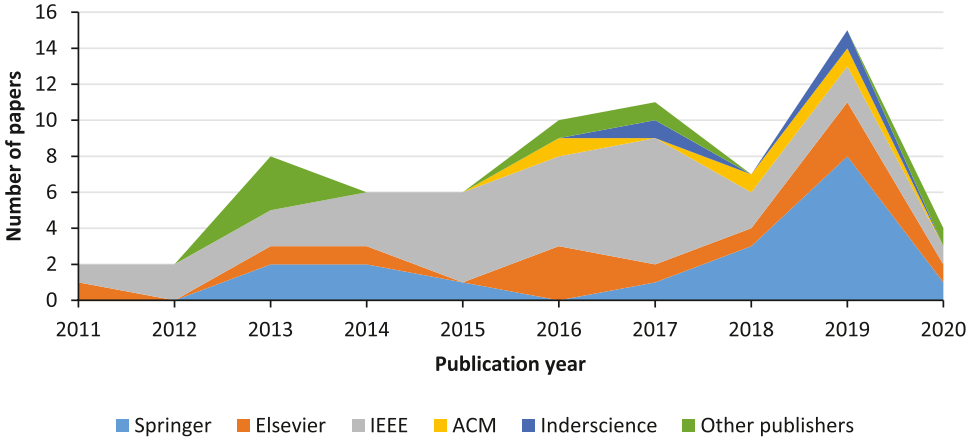


Fig. 1. Year-wise collected research papers on meta-heuristics based cloud task scheduling.

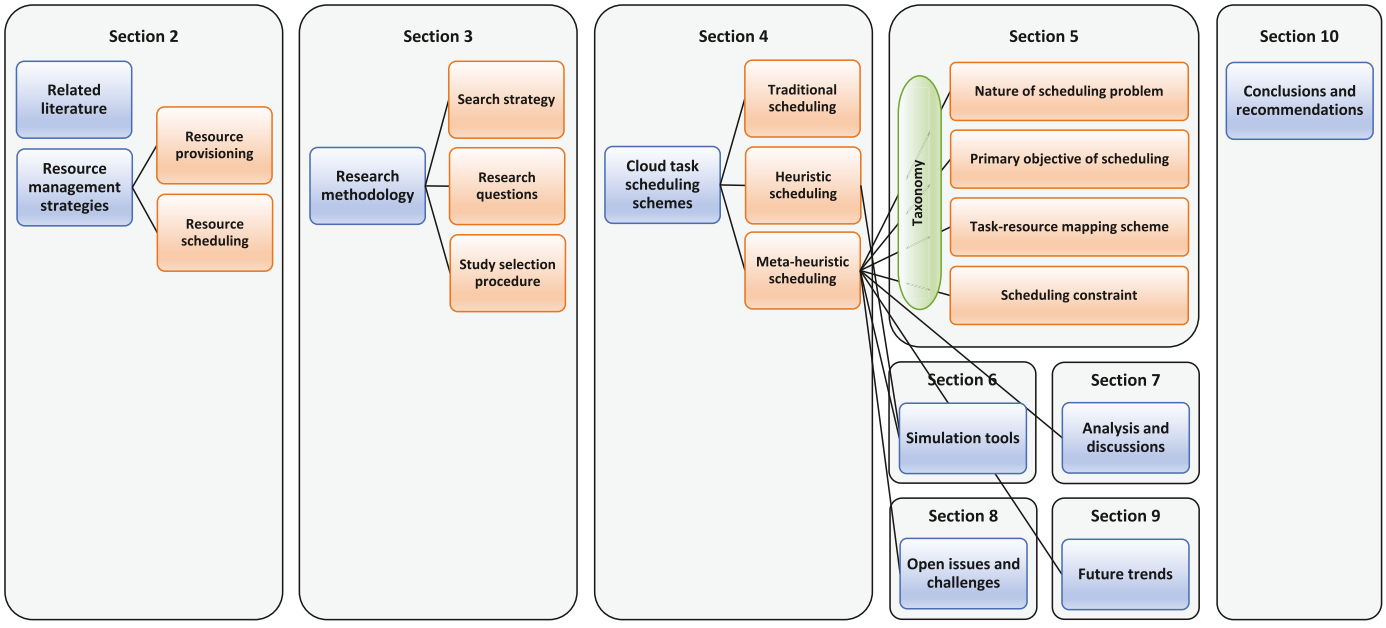


Fig. 2. The structure of the study.

3. Benefits and limitations/weaknesses of each state-of-the-art or emerging meta-heuristic or hybrid scheduling approach are outlined.
4. The performance of existing approaches is assessed, based on qualitative QoS parameters-based metrics.
5. Various simulation tools widely used in cloud are also presented and compared.
6. Findings pertaining to the involved aspects are verified through an in-depth analysis and discussions.
7. Open issues are put forward, and future works suggested in previous researches are summarized, which might help draw a roadmap for researches on current trends and potential future research directions.

1.3. Organization of the paper

This paper is structured as: Section 2 provides a brief, albeit informative overview of the relevant literature as well as a comprehensive foundation of the common resource management strategies in cloud. The research methodology followed in this study is described in Section 3, including source of data, search criteria, study selection method, etc. Section 4 begins with a brief introduction to traditional and heuristic scheduling, followed by a taxonomy of well-known meta-heuristic scheduling algorithms in cloud computing to interlate meta-heuristic scheduling algorithms with their predecessors to reduce their learning effort, especially for those newbies to the field. A novel taxonomy of meta-heuristic task scheduling techniques in cloud environments, along with their pros and cons, is introduced in Section 5. Section 6 discusses a comparison between simulation tools frequently used for evaluating novel cloud scheduling techniques. Thorough analysis and discussion of the data-rich key findings of this review, along with its strengths and limitations, is presented in Section 7. Open issues and challenges as well as potential future trends on cloud scheduling are highlighted in Sections 8 and 9, respectively. Finally, conclusion of the paper is drawn in Section 10.

2. Preliminaries

2.1. Related literature

For several recent years, scheduling of tasks in cloud computing has been a fertile research ground. Thus, there is an enormous number of surveys and papers on solutions related to this area in the literature. In what follows, a few related works are discussed in two main respects: i) surveys that aim at exploring the updation on scheduling of cloud tasks based on meta-heuristics, and ii) seminal works by which the novel taxonomy introduced in this paper is inspired. It should be also pointed out that through in the following subsections, our taxonomy proposal is always being put to a peer with other taxonomies in the literature, in order to emphasize the distinction of this study, compared to others.

2.1.1. Relevant surveys

In cloud environment, there is an ongoing much research as some critical research challenges are still overlooked, such as scheduling of applications, resource provisioning, load balancing, energy consumption, etc. Research on task scheduling in cloud is still in its early stages, so further improvements are needed. In this subsection, some review papers are discussed regarding meta-heuristic task scheduling techniques useful for the present survey as well as relevant to our research. Although this section discusses the related survey papers in the task scheduling domain, these surveys neither consider nor classify the vast majority of meta-heuristic scheduling techniques. Most of them either address the meta-heuristic techniques subsidiarily or undertake a limited category of them (i.e., nature-inspired or bio-inspired).

A cloud resource broker (scheduler) operates as an intermediary between the provider and users of the cloud service. It significantly helps tackle the portability and interoperability challenges that hinder sharing resources among interconnected clouds in cloud computing. In a comprehensive manner, Chauhan et al. [25] analyzed diverse cloud brokering techniques by presenting them in the form of a taxonomy, highlighting their strengths and weaknesses/limitations including trust techniques, optimization, QoS parameters, pricing, and multi-criteria. These techniques are evaluated by measuring different performance metrics. The goal of measuring such typical metrics is similar to our goal in this paper; however, they neither consider meta-heuristic techniques nor define a rigorous taxonomy that can include diverse scheduling problems in clouds.

In [26], an extensive review of various meta-heuristic approaches was introduced, where optimal load balancing solutions are presented by these approaches in both grid and cloud systems. Furthermore, these techniques mainly focus on makespan, waiting time, and response time, as well as factors of energy consumption and carbon emitting. However, this study does not trigger really major issues in cloud computing, like server consolidation, energy management, VM migration, resource scheduling, SLA, etc. Moreover, no general or formal taxonomy is defined in [26].

Finally, there are many other relevant surveys, each considering only a certain aspect of meta-heuristic scheduling techniques in cloud. For instance, a literature review of swarm intelligence based scheduling solutions in cloud computing is found in [27], while a survey of PSO-based scheduling approaches in cloud computing was conducted in [28], which also lack comprehensiveness.

2.1.2. Inspirational taxonomies

In fact, the novel taxonomy proposed in this study is broadly organized into three parts: one categorizes existing well-known meta-heuristic algorithms widely applied to cloud scheduling problem into swarm, evolutionary, physical, emerging, and hybrid algorithms, as per Section 4; while the second part describes diverse meta-heuristic scheduling approaches in cloud computing in terms of the nature of scheduling problem (i.e., single- or multi-objective), the primary objective of scheduling (QoS parameters), the task-resource mapping scheme, and the scheduling constraint; finally, existing open challenges as well as potential future research in cloud computing are inferred in the third part, in order to shape the direction to improving task scheduling in clouds based on meta-heuristic optimization algorithms.

Regarding the taxonomy of meta-heuristic scheduling approaches in Section 5, it is mainly inspired by the scheduling theory introduced in [29] and reviewed in a few books [30,31]. A scheduling problem should, according to this theory, be defined in terms of the operational model, the optimality criterion, the resource allocation mechanism, and high-quality constraints. Our categories, respectively, the scheduling problem, QoS requirements, resource allocation criteria, and scheduling constraints are triggered by this idea. These notations are existing for decades now, in order to tackle deterministic scheduling problems [32]; however, the application of these notations to scheduling problems in modern cloud computing systems is difficult and unnatural because, in these systems, the important features that fully characterize scheduling problems cannot be modelled using these notations. Some examples of such features are scheduling level, workload source and structure, resource scaling and sharing, and the adaptability requirement.

Thus, our taxonomy has been undergone further enhancement, driven by the taxonomy defined in [33] to maintain the general purpose of cloud computing systems by emphasizing the innovative scheduling solutions. The main reason behind refining that taxonomy is that over the last thirty years, current applications and their environments become significantly more complex. In addition, the taxonomy in [33] has been extended in our work into four aspects. First, the notion of single and multi-objective assignments is introduced, which is indeed of much importance, especially with the existence of both limited resources and scalability issues. Second, Thakur and Goraya [33] considered more generalized techniques: i) swarm-based, ii) evolution-based, and iii) statistics-based algorithms, which are adopted to collect and analyze feedback statistical data about energy consumption, resource utilization, performance, etc., for optimal distribution of the workload. In our work, the techniques i) and ii) have been replaced by other respects which are focusing more both on task-resource mapping schemes and scheduling constraints. However, we have found that those aspects are important and should be further deepened to include artificial intelligence-based scheduling methods and other kinds of prediction-based resource allocation as well. Due to its scarcity in cloud task scheduling applications, the technique iii) has been omitted, addressing rather physics-based approaches. To be mentioned, the differentiation between sub-optimal (i.e., heuristic, meta-heuristic, etc.) and approximate (i.e., traditional) solutions, which was covered in [33], has not been addressed in our study, for the sake of simplicity.

Finally, the last part of our broad taxonomy meant for the description of open challenges and recent research directions is identified as per Sections 8 and 9, inspired by the taxonomy proposed by Sing and Chana [34]. In their research, 110 research papers from the existing literature were studied, and only 71 out of them were recognized as the most relevant papers to scheduling techniques in cloud computing. In this study, various ways including the classification of resources as well as tracking the evolution of task scheduling were adopted to examine the consequences. As per the research questions, the results were analyzed based on a detailed classification of the task scheduling techniques, their subtypes, their usage ratios, as well as the desired QoS requirements. Moreover, comparisons of task scheduling algorithms, task scheduling tools, and resource distribution policies were conducted. Moreover, existing research work is summarized in a systematic manner, and a broad methodical literature analysis is provided on the cloud area with regard to resource allocation in general and resource scheduling in particular, aiming to reveal the vigorous research gaps to be bridged in future research. However, the taxonomy proposed in this study is too broad and forked, which requires more time and concentration from readers to audit about the involved existing challenges and their potential solutions. Therefore, our suggested taxonomy in this regard is defined to be more simple, specific, and well-organized to categorise and summarize existing task scheduling problems and their potential solutions.

2.1.3. Related taxonomies

Over the last ten years, researchers and scholars proposed many different, sometimes overlapping taxonomies, each targeting a specific aspect of cloud computing. Cloud task scheduling was addressed in some particular taxonomies based on nature-inspired techniques [35,36], bio-inspired techniques [37,38], QoS parameters-based techniques [39,40], and cost optimization techniques [41]. There is a common problem in these taxonomies; that is, they lack more generalization or comprehensiveness of scheduling techniques in cloud computing systems. To the best of our knowledge, there is no existing taxonomy that may include diverse meta-heuristic scheduling techniques available in the literature. Moreover, the aforementioned taxonomies were proposed specifically to survey the state-of-the-art; however, their authors did not manage to include related scheduling techniques. To say the least, no comprehensive type of taxonomy is existing in the literature to describe diverse scheduling techniques in the cloud clearly, rigorously too.

In [42], a comparative review, taxonomy, and comprehensive survey of cloud workflow scheduling techniques were conducted in a problem-solution manner. Some research directions were also highlighted in this study for future further investigation. In this research work, the scheduling problem is surveyed in three dimensions: robust, elastic, and economic scheduling, thus ignoring some vital QoS parameters, such as makespan, energy consumption, security, etc. Moreover, the limitations of the involved scheduling techniques are not addressed in this study.

In a study [43], scheduling problems and solutions were presented in a generalized taxonomy of distributed systems. This taxonomy is specially designed to classify 109 scheduling problems along with their solutions. The features of a taxonomy are utilized to further cluster those 109 problems into ten groups. The taxonomy proposed in this study aims to minimize redundant efforts from the researchers by increasing new research visibility based on the prior art. However, there are still several challenges and limitations omitted. For example, imprecise estimation of resource requirements may not be tolerated by existing management policies, calling for managing to propose new trade-offs between the optimality of a policy and its ability to handle insufficient information about incoming workloads [44].

In another research work by [45], 91 papers on scheduling algorithms were reviewed, 23 out of which are discussing meta-heuristic approaches. Various issues, like QoS parameters-based allocation techniques, static and dynamic allocation strategy, etc., are discussed in this work based on meta-heuristic algorithms. Broader, systematic comparison between diverse meta-heuristic algorithms as well as a comprehensive review of them could be beneficial as well.

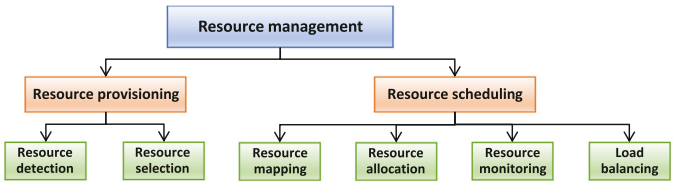


Fig. 3. Resource management in cloud environment.

A taxonomy of existing research into maintaining cloud resources for optimizing the execution of Bag-of-Task (BoT) applications was presented in [46]. There are multiple independent tasks in BoT applications, each executed in any order on a VM. Both the commercial organizations and scientific communities frequently use this type of application. Indeed, our taxonomy does not explicitly model the requirements to operate this optimization task. When necessary, a special taxonomy must be developed to act as a complement. However, our taxonomy alludes to BoT operations by means of elasticity and scalability metrics, which is very helpful for identifying scheduling problems which might need further resources.

Finally, other different taxonomies were presented, in which the entire spectrum of scheduling techniques is not fully covered. For example, [35] presented a qualitative review of all techniques included, considering the key methods, the main goal, advantages, and limitations. These taxonomies involve a comparison between nature-inspired algorithms using their features to quantify their level of energy efficiency and utilization of resources, two similarities to our taxonomy. Another cloud-related taxonomy was depicted in [39] in order to present the scheduling solutions by classifying them based on only one of two alternatives: QoS-constrained or best-effort. QoS-constrained scheduling in [39] aims to minimize makespan under QoS constraints, like security, load balancing, etc., while best-effort refers to solutions that target the minimization of makespan and cost without considering other constraints or goals. Similar terms are used in our taxonomy to characterize the scheduling solutions. Moreover, in the taxonomy proposed by [37], the solutions are characterized as being bio-inspired. While an expanded set of technique types is covered in our taxonomy with a higher rigorousness.

2.2. Resource management strategies

As shown in Fig. 3, different resource management strategies in clouds can be grouped into two basic paradigms: resource provisioning and resource scheduling [47]. Resource scheduling is further categorized as per resource allocation, load balancing, resource monitoring, and resource mapping. In the following, the basic idea behind resource provisioning techniques is first discussed since, in the cloud computing paradigm, they came prior to scheduling techniques. Thereafter, the basic concepts of scheduling techniques are taken up, along with the corresponding motivations.

2.2.1. Resource provisioning

Resource provisioning is defined as the act of allocating virtualized resources to users. When a user's request for resources is accepted by the cloud service provider, a convenient number of VMs is created and allocated, as per demand, to that user using a resource provisioning technique. Moreover, it is the main responsibility of resource provisioning to ensure the fulfilment of the SLA negotiations and users' needs based on QoS requirements as well as map the incoming workloads or applications (cloudlets) to resources (VMs) [48].

Resource provisioning mainly aims to estimate the scale of an upcoming workload/application request (demand) in order to select the

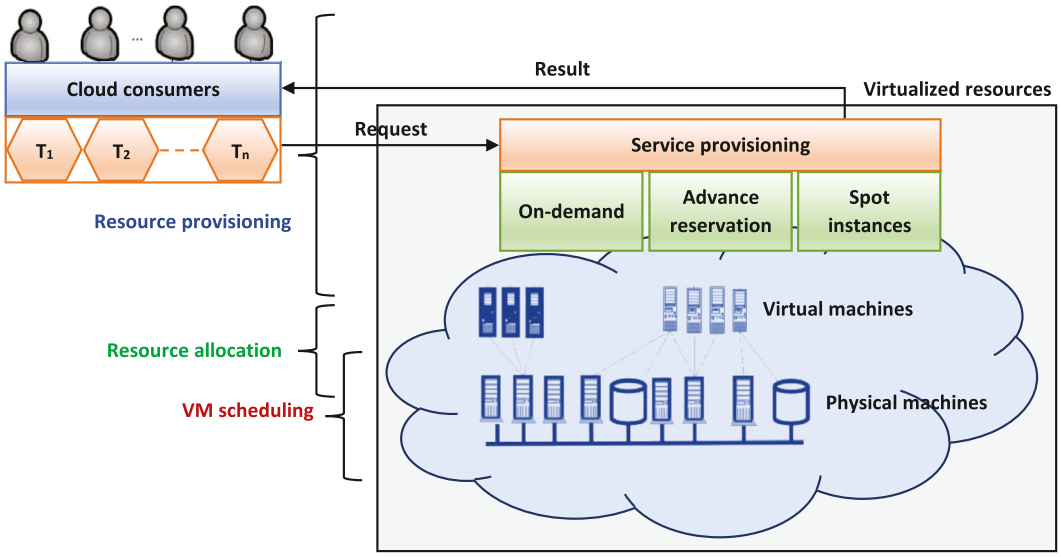


Fig. 4. Resource provisioning plans.

best resources for the submitted user and allocate them to the that workload. In other words, incoming tasks/applications should be served with as minimum amount of resources as possible while maintaining a desirable QoS level (maximum resource utilization). Incoming request is mapped to running VMs using a resource provisioning approach so that the services can be handled in minimum time and at the lowest cost (maximum throughput and maximum savings) while the highest profit is generated to the service provider, without violating the SLA, too [49]. As shown in Fig. 4, resource provisioning techniques are majorly provided to the cloud consumers in three basic forms:

- **On-demand provisioning:** It allows users to pay per resources being used based on an intermediate level plan; that is, at a given time t, if the demand for available cloud resources transcends a specified value, then on-demand resource provisioning is evoked for providing additional resources. On-demand plan can be also used to solve the underprovisioning problem through carving out more resources at an additional cost. In general, allocating on-demand additional resources requires more cost than allocating advance-reserved resources to users [50].
- **Advance reservation:** On the basis of a long-term plan, this mechanism enables the cloud users to pre-reserve resources for a specified period of time. This scheme is very useful for both the Elastic Compute cloud (EC2) and the federated cloud; however, it has some challenging issues including prediction of future demand and prices of cloud resources. The underprovisioning and overprovisioning types of problem might also arise in this approach [51].
- **Spot instances:** On the basis of a short-term plan, this mechanism allows customers to tender unused resources based on the Amazon's third plan in which unused resources are offered as spot instances at a much lower cost than on-demand provisioning and advance reservation. Amazon Web Services (AWS), Google Cloud, and Microsoft Azure, being the major cloud service providers, make this scheme available to users. The main demerit of spot instance techniques is the frequent variation of resources price rate based on supply and demand [52].

Advantages of cloud resource provisioning: Resource provisioning brings about numerous favors to the cloud, some of which are highlighted below [53]:

- Efficient resource provisioning techniques reduce makespan time and response time for submitted workloads.
- The issues of overprovisioning and underprovisioning can be reduced through the optimized utilization of resources.
- Better resource provisioning can be brought to cloud environments through reducing VMs' startup delay.
- Both the robustness and fault tolerance capabilities can be brought using effective cloud resource provisioning algorithms.
- Power consumption can be also reduced using a resource provisioning algorithm without violating the SLA.

2.2.2. *Resource scheduling*

Scheduling is the art of analyzing the required QoS parameters with the aim to determine which activity should be performed. In clouds, scheduling is responsible for: i) selecting the most optimal VM to execute a task using a heuristic/meta-heuristic algorithm, and ii) ensuring the fulfillment of QoS constraints. In general, there are two ways of resource (task) scheduling. The first method is "on-demand scheduling" in which the resources are quickly provided for random workloads by cloud service providers. In this manner, the problem of workload dispersal may be raised (i.e., a single VM might process multiple tasks at a time). Consequently, performance degradation might be encountered, leading to the overprovisioning sort of problem. "Long-term reservation" is the second method in which the underprovisioning type of problem might occur due to the ideal circumstance posed as a result of providing a larger number of VMs. Unnecessary wastage of time and resources might raise the overprovisioning and underprovisioning problems that in turn increase the cost of services. To handle this sort of problem, an efficient resource provisioning technique is required to efficiently analyze and schedule the submitted workloads. Fig. 5 depicts the process of Resource Provisioning with Scheduling (RPS) [53].

The overall aim of RPS is to provision the VMs to users in such a way that: i) fulfills the cloud consumers' demand without violating the SLA, and ii) enhances the proactive understanding of the expectations and requirements of those consumers based on the size of incoming workloads. As shown in Fig. 5, after a proper analysis of incoming workloads, an SLA commitment is concluded between the service provider and the cloud consumer. For each workload, the required QoS parameters are identified to calculate the Fitness Function (FF_{QoS}) which is self-compared in the case that QoS parameters are not considered ($FF_{non\text{-}QoS}$). After that, the condition is checked (assuming a minimization problem): if

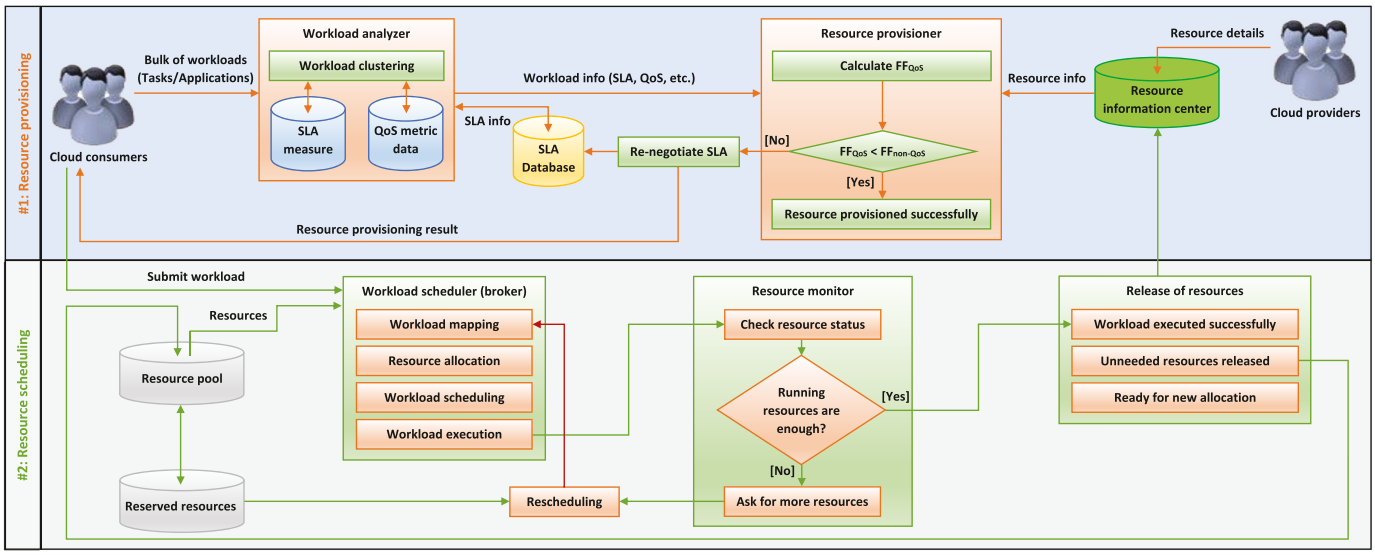


Fig. 5. Provisioning and scheduling of cloud resources.

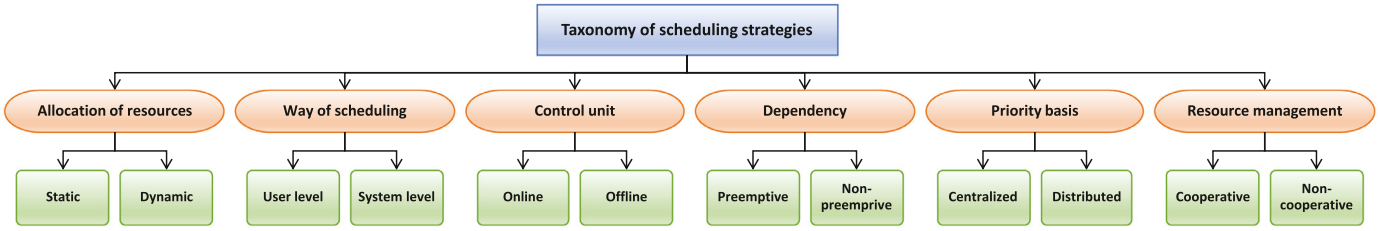


Fig. 6. Different strategies of scheduling in clouds.

the FF_{QoS} value is less than the $FF_{non\text{-}QoS}$ value, then the provisioning process is performed; otherwise, the workload is analyzed again after the cloud consumer resubmits the SLA through renegotiation.

As shown in Fig. 5, if the resource provisioning process is successfully completed, a scheduling algorithm is adopted for processing the incoming workload within a predefined deadline and/or budget. The load at each resource is estimated regularly through monitoring the running resources (VMs) before making a mapping (scheduling) decision of incoming tasks/applications (workload) to those resources. If a VM becomes overutilized, then its resources are temporarily excluded from any future allocation. Incoming workload is mapped to available resources and the condition if a running VM has the capability or not to execute the workload is tested. Based on that, the resources are allocated to the workload if the running resources are sufficient, and the QoS requirements are then evaluated; otherwise, the horizontal scalability theory is applied in order to extend the available resources.

Based on the concepts of provisioning and scheduling of cloud resources, various research papers (year-wise) in this regard were collected to be reviewed in this study. Cloud computing has various types of scheduling algorithms that can be categorized in terms of: static and dynamic, user level and system level, online and offline (batch mode), preemptive and non-preemptive, etc. However, scheduling algorithms are mainly divided into two classes: static and dynamic. Various scheduling strategies in clouds are outlined in Fig. 6.

- **Static and dynamic scheduling:** Static scheduling algorithms require advance information about incoming tasks (i.e., task count, task length, task deadline, etc.) and available resources (memory, processing power, node processing capacity, etc.). Static algorithms perform better when the workload is not varying frequently and the variation in system behavior is very little. In cloud environment, load fluctuates constantly; so static algorithms are arguably inadequate for cloud computing. To implement a static algorithm is very easy; however, in these type of algorithm, the QoS parameters are not duly optimized, as well as high performance can be absent in the real environment [54]. Therefore, the cloud environment was in need of new convenient dynamic scheduling algorithms. First In First Out (FIFO [55]), Shortest Job First (SJF [56]), Round Robin (RR [57]), etc., are a few examples of static algorithms. Contrariwise, dynamic algorithms need no advance information about the node (VM) and task; however, the node requires continuous monitoring. In cloud environment, these algorithms are more accurate, efficient, and suitable as when any node becomes overloaded, the task being executed on this node can be then consciously transferred to an underloaded node (i.e., as the load changes at a node, whether it increases or decreases, the algorithm behavior changes dramatically) [58]. Dynamic RR [59], Clustering Based Heterogeneity with Duplication (CBHD [60]), Heterogeneous Earliest Finish Time (HEFT [61]), Weighted Least Connection (WLC [62]), and meta-heuristics like Ant Colony Optimization (ACO [63]), Particle Swarm Optimization (PSO [64,65]), etc., are a few examples of dynamic scheduling algorithms widely applied in cloud environments. As shown in Table 2, both static and dynamic algorithms have their valuable advantages as well as some unavoidable disadvantages. As per the literature, dynamic task scheduling algorithm is highly recommended for cloud environment due to the frequently oscillating nature of workloads and system behavior in cloud computing. Hence, dynamic algorithms, including meta-heuristics, can effectively help imple-

Table 2
Comparisons between static and dynamic algorithms.

Comparison	Static algorithms	Dynamic algorithms
Need an advance information about the incoming requests/tasks?	Yes	No
When a scheduling decision is made?	At compile time	At runtime
How complexity is the implementation?	Low	High
Achieve optimal results for large-scale optimization scenarios?	No	Possible
How long time needed for solving computational problems?	Long time	Short time
Which types of algorithms come under both types?	Traditional algorithms	Meta-heuristic algorithms
Finding out an optimal solution for multi-objective problems?	Difficult	Easy
Which workload type is adequate?	Static workload	Dynamic workload
Efficient workload balancing on the running VMs (nodes)?	No	Potential
Continuous monitoring of nodes?	No	Yes

ment approximate solutions to NP-hard problems such as cloud task scheduling.

- **User level and system level scheduling:** In user level scheduling, the problem of service provisioning between service providers and consumers is tackled by the scheduler, which shows a high efficiency, especially when market-oriented virtualized resources are delivered as a service to users. On the other hand in system level scheduling, the resource management is taken up by the datacentres, that in turn impacts the performance of the datacentre appreciably.
- **Online (immediate) and offline (batch) scheduling:** With regard to online scheduling, the scheduler maps the customer request to running VMs such that the scheduling process is kept stable over time by performing a single scheduling for each task at a time. Minimum Completion Time (MCT [66]), Minimum Execution Time (MET [67]), Opportunistic Load Balancing (OLB [68]), etc., are a few examples of online-mode scheduling algorithms. As for offline scheduling that is also called batch mode scheduling, resources are allocated in response to incoming application request, based on predefined moments, which is very useful for rapid calculation of the processing time when there is a larger number of incoming tasks. Min-Min [69], Max-Min [70], etc., are a few examples of batch-mode scheduling algorithms.
- **Preemptive and non-preemptive scheduling:** With regard to the preemptive scheduling, the tasks currently being executed could be interrupted and consequently are properly migrated to other free resources. While in non-preemptive scheduling, cloud resources currently allocated to a task can be only released when the task execution is successfully completed (i.e., task execution happens completely at the resource without interruption). In clouds, one resource is used at a time to execute only one task (i.e., the referred interruption will never occur while executing tasks in cloud environments) [71].
- **Centralized and distributed scheduling:** In centralized scheduling, all tasks are gathered by a master processor unit and then are sent to slave processing units where every processor takes over a single dispatch queue [72]. On the other hand, there is no central control unit in distributed scheduling as local schedulers are responsible for: handling the incoming requests, as well as maintaining the status of all other processors by continuously sharing updates with them [73].
- **Cooperative and non-cooperative scheduling:** In cooperative scheduling, all processors achieve their common goal through collaboration when making a scheduling decision [74]; that is, a cooperative scheduler uses a system tick created by a periodic timer to schedule tasks which are later in executed sequentially through the synergy of processors. In non-cooperative scheduling, every individual processor makes its decision independently while other processors are not affected or ever alerted [74].

Need for scheduling: Scheduling mainly aims to handle end-users' incoming requests by finding out the best cloud resources that should improve both the resource utilization rate and key performance parameters (QoS parameters) [34]. Cloud computing has diverse performance indicators, like makespan, monetary cost, execution cost, response time, energy consumption, reliability, etc. An efficient task scheduling algorithm must be used to analyze and improve these parameters in order to fulfill the requirements of both end-users and service providers without violating the SLA. Existing scheduling algorithms cannot resolve such problems due to obstacles, such as dispersion of resources, dynamism, and heterogeneity. Thus, a scheduling algorithm is needed for equitable and proper distribution of heterogeneous workloads across VMs based on the capacity of resources, with the overall aim to overcome the potential problems of overloading and underloading in cloud task scheduling.

3. Research methodology

For a deeper understanding of task scheduling techniques based on meta-heuristics, this section presents a rigorous literature survey as well as research guidelines on collecting meta-heuristics based cloud scheduling approaches whose objective is to optimize QoS parameters in accordance with cloud consumers' requirements. As per the survey [47], sources of data, search strategy, research questions, and study selection method should be thoroughly considered for creating an efficient research methodology more oriented to cloud task scheduling. Here, it should be mentioned that the systematic process followed in this study is to provide researchers with more transparency in the cloud area as well as to facilitate the potential development of new algorithms for scheduling of applications in cloud environments.

3.1. Sources of data

A variety of reputed scientific databases have been inspected, including Scopus, Web of Science (WoS), IEEE Xplore, ACM Digital Library, Springer Link, Google Scholar, etc., and many useful, relevant research papers were subsequently found regarding meta-heuristic scheduling techniques in the field of cloud computing. Moreover, a list of different research questions, along with the motivations behind them, is illustrated in Table 3, which was eventually answered by first defining the primary studies, then applying the inclusion/exclusion criteria, and evaluating the results at the end, which is explained, in detail, through in the following subsections.

3.2. Search strategy

Our research started in January 2020 and research papers related to meta-heuristic scheduling techniques in clouds were explored using a combination of main keywords, like meta-heuristics, meta-heuristic optimization, and cloud, with other keywords, such as resource scheduling, resource allocation, task scheduling, workflow scheduling, load balancing, SLA-based scheduling, QoS parameters-based scheduling, single-, and multi-objective scheduling techniques in cloud computing. After extensive tests, we finally decided on the following search query, by which a plenty of relevant quantitative and qualitative research studies have been discovered:

- ((“task scheduling” OR “resource scheduling”) AND (“meta-heuristic optimization” OR “meta-heuristics”) AND (“cloud” OR “cloud computing”)).

Indeed, basic research into the scheduling issue in clouds has actually started in 2005, but after the mid of 2008, unruly development

Table 3
Research questions and motivations.

No.	Question	Motivation
RQ1	How much progress is task scheduling based on meta-heuristics?	Cloud computing has lots of proposed meta-heuristic scheduling algorithms which are reported in the introduction, preliminaries, and taxonomy parts of this paper.
RQ2	How meta-heuristics differ from traditional and heuristic algorithms?	Meta-heuristics are characterized by distinctive characteristics in performance and accuracy over traditional and heuristic algorithms, as per Section 4.
RQ3	How the time and cost parameters are optimized at the same time, with the existence of deadline or budget as a constraint?	Both the time and cost parameters are difficult to optimize simultaneously.
RQ4	What is the best workload-VM allocation scheme to overcome the issues of overloading and underloading?	Meta-heuristic scheduling algorithms are divided in the present study into four broad categories, taking a variety of aspects into account (e.g., nature of the scheduling problem, primary scheduling objective, task-resource mapping scheme, and the scheduling constraint).
RQ5	How to implement an efficient meta-heuristic task scheduling approach for improving user-defined QoS parameters?	Scheduling criteria and their success to properly optimize QoS parameters are used as a strong performance metric to diagnose different scheduling algorithms, revealing as well some open research challenges that would promote potential future research.
RQ6	Why meta-heuristics are a preferred choice for task scheduling in cloud environment?	
RQ7	How the main QoS parameters are effectively optimized using the existent meta-heuristic task scheduling algorithms?	
RQ8	How the SLA changes with respect to time?	It is crucial to determine the extent to which the QoS parameters are fulfilled and cloud resources are sufficient, when executing different applications.
RQ9	What are the QoS parameters and resources most concerned with in the cloud task scheduling area?	
RQ10	Which meta-heuristic techniques can accomplish the task scheduling process most efficiently?	This study mainly aims to acknowledge the effective role that each meta-heuristic approach in the litterateur has played to improve cloud task scheduling, based on analysis and discussion.
RQ11	What is the simulation tool widely used for performing task scheduling experiments in cloud computing?	The cloud computing field is supported with numerous versatile simulation tools used for performing the experiments.
RQ12	Which research gaps are still unaddressed in meta-heuristic task scheduling approaches?	One of the major aims of this review is to help researchers understand deeply both the current open issues and future requirements regarding task scheduling in cloud computing.
RQ13	What are the current trends on meta-heuristics based scheduling most accessed and exciting for future research?	The majority of current research is geared towards green computing and other promising trends in the field.

has taken place. Therefore, the search window was selected within the period from 2011 to 2020 for spotting the relevant papers on meta-heuristic based scheduling of applications, keeping in mind that the collected research papers are published in reputed journals and conferences (e.g., IEEE, Springer, Elsevier, ACM, Inderscience, etc.). In this work, a quick search strategy was applied where lots of recently published papers (2017-2020) are emphasized in order to make the study well-intentioned and up-to-date for the cloud research community. After that, publications from 2011 to 2020 were considered as overall.

3.3. Research questions and their motivations

Research Questions (RQs) adopted in this study and pertaining to meta-heuristics based task scheduling approaches in clouds, along with the corresponding motivations, are outlined in Table 3.

Table 4
Inclusion/exclusion criteria to select the papers.

Inclusion criteria	Exclusion criteria
Task scheduling in cloud computing is mainly addressed in the selected study.	Other resource management issues are not fully considered in the selected study.
At least one meta-heuristic technique for task scheduling is considered in the selected study.	The selected study does not consider meta-heuristic techniques at all.
The language of the selected study is English only.	The language of the selected study is not English.
The selected study is published in scholarly society as well as peer-reviewed.	The selected study is not subject to a peer-review, like technical reports, descriptions, and workshops.
Publication of the selected study is within a well-reputed conference or journal.	Publication of the selected study is not in the form of books, abstracts, articles, keynotes, or editorials.

3.4. Study selection procedure

As shown in Fig. 7, the study selection criteria followed in this study was executed by going through several reputed journals hosted by IEEE, Elsevier, Springer, Wiley, etc., and collecting various research papers by using the keywords aforementioned in Section 3.2. Primarily, after reading the title of the articles, most research papers were excluded because their titles are not conforming to our present survey. After that, investigating their abstracts and conclusions, some research papers were also omitted.

As per Fig. 7, initially, 761 research papers were collected, but based on the criteria identified and after a purposeful study, most articles were screened out. All remaining papers (254) were undergone further analysis, and full body of some articles were found irrelevant to our main topic; therefore, they were also excluded. Keywords mentioned above in Section 3.2 as well as inclusion/exclusion criteria defined in Table 4 were used to investigate the selected papers and finally, 71 research papers were settled on for our survey as shown in Fig. 7. In fact, there are many available pertinent research papers in the literature of meta-heuristics based cloud task scheduling; however, selected papers in this research were carefully picked from within various reputed publishers, like Springer (conferences and journals), Elsevier (conferences and journals), IEEE (conferences, journals, and transactions), ACM (conferences and journals), Inderscience (journals), etc., from 2011 to 2020, as depicted in Fig. 1.

All the collected papers were closely analyzed as shown in Fig. 1, and it has been found that most papers are published after 2012. Fig. 1 also reveals the number of selected papers, in which an exponential growth in research is significantly observed in the field of meta-heuristics based cloud task scheduling from 2013 to 2020. It is also noticed that the

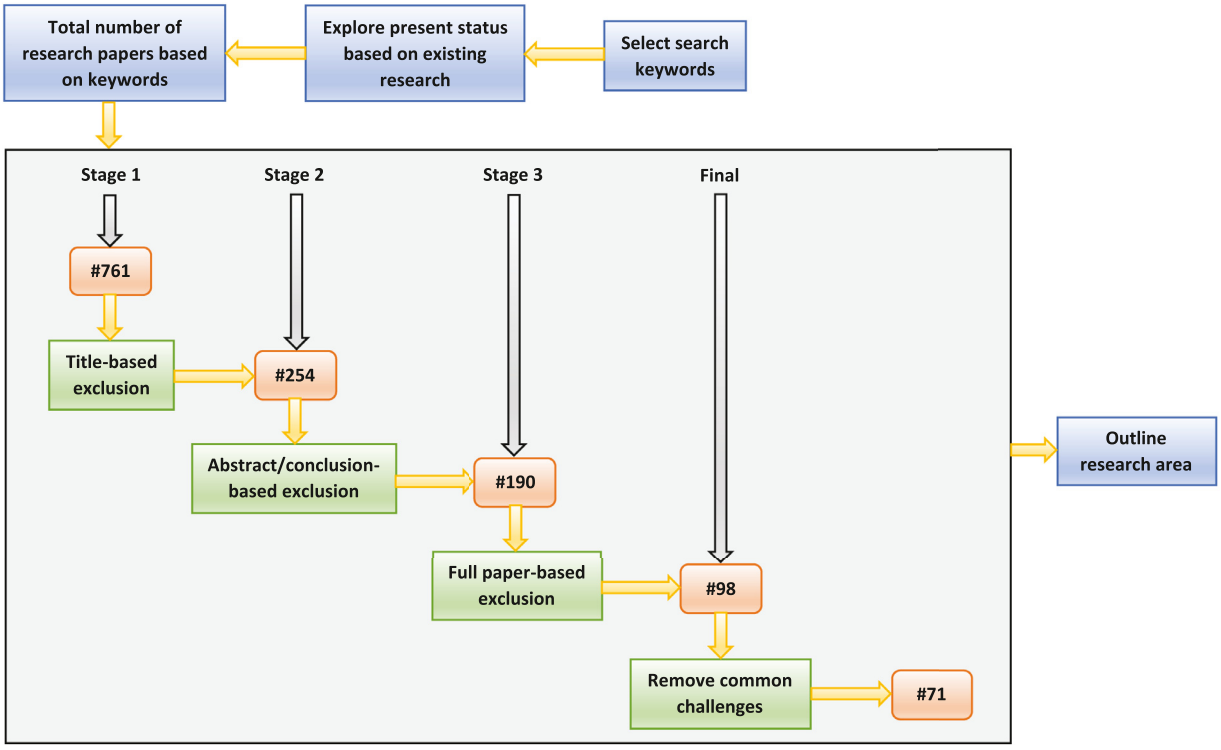


Fig. 7. Study selection process.

majority of these papers are published in IEEE and Springer as illustrated in Fig. 1.

4. Categorization of cloud task scheduling schemes

4.1. Traditional scheduling

So-called task scheduling can be expressed as a problem in which a given set of n tasks $\{T_1, T_2, \dots, T_n\}$ are to be assigned to a range of available m machines $\{M_1, M_2, \dots, M_m\}$, on which those tasks will run, taking into account the optimization of one or more predefined objective functions or measures. On the one hand, when $m = 1$, (i.e., one and only one machine exists), the scheduling problem is then called a single-processor (single machine) scheduling problem. On the other hand, when $m > 1$, (i.e., more than one machine exists), the scheduling problem is called a multi-processor (parallel machine) scheduling problem. For evaluating the performance of different scheduling algorithms, their variants as well as multiple objective functions including makespan, flowtime, tardiness, and lateness, have been widely applied in the scheduling community. For further measurement methods, [1] and [10] are two helpful citations to readers.

To make the idea more concrete, a simple example is illustrated in Fig. 8 where a set of six tasks is given and the results of their allocation to both a single and parallel machine are revealed, along with the completion time (c_j) and due day (d_j) of each task as shown in Fig. 8(a). Speaking of a single-processor (single machine), Fig. 8(b) shows the results; that is, in the case that $m = 1$ and $n = 6$, if makespan is defined as the objective function, then the solution $(1, 6, 5, 2, 4, 3)$ gives a completion time $c_{\max} = 12$; otherwise, if the objective function is reformulated to indicate (represent) the number of tardy/late tasks (jobs that did not meet their due time), then $U_{tot} = 1$ for the same solution. In the case of parallel machines, Fig. 8(c) gives the results where, if the objective function of makespan is adopted while $m = 2$, then the solution $(1, 6, 5, 2, 4, 3)$ gives $c_{\max} = 7$. However, Fig. 9(a) reveals that more improvement in $c_{\max}$ is possible, saying that the total completion time (c_j) of all the incoming tasks (n) is 12 and $m = 2$, a potential $c_{\max}$ is equal to $(12/2 = 6)$. Based on the adjustments of the combinations (T_4 and T_3) attempted by most studies on the critical path (T_1, T_4, and T_3) [75], Fig. 9(b) gives an example that illustrates a more potential reduction in $c_{\max}$ from 7 to 6. These diverse scenarios emphasizes that a given problem can be made to conform to any case in question by adapting the given objective function and constraints.

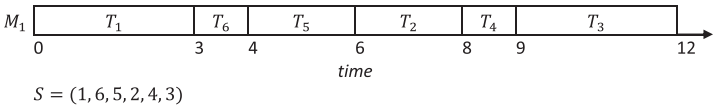
Many problem domains are widely applying the scheduling paradigm. In terms of traditional scheduling, a taxonomy of problems was technically presented in several studies [3,76,77] with respect to the features of tasks (i.e., weight, due date, release date, and processing time), machines (i.e., single or multiple), as well as many other details, such as online vs. offline, batch vs. non-batch, precedence vs. non-precedence, sequence-dependent vs. sequence-independent, preemption vs. non-preemption, etc. Typically, scheduling problems are differentiated and described by employing these constraints. All scheduling problems addressed in [77] were presented based on a three-fold notation, $\alpha/\beta/\gamma$. In this notation, the machine type is described by α (i.e., single or parallel machine), the processing characteristics and constraints are determined by β (i.e., sequence-dependent or sequence-independent), and finally, the value of measures is referred to by γ (e.g., makespan or number of late jobs). In [76], scheduling was classified based on the length of scheduling time into several levels: short-range, middle-range, long-range planning/scheduling, and reactive control/scheduling. Furthermore, in [1], a large amount of details about traditional scheduling algorithms are presented in a comprehensive survey replete with constructive comments from various perspectives.

4.2. Heuristic scheduling

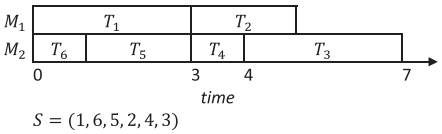
Here, we do not need to be reminded that the main purpose of this study is to address meta-heuristic scheduling algorithms in cloud computing. However, it would be so helpful if we delve a bit deeper into

	T_1	T_2	T_3	T_4	T_5	T_6
Due day (d_j)	3	8	11	12	6	4
Completion time (C_j)	3	2	3	1	2	1

(a) Each task's due day and completion time.

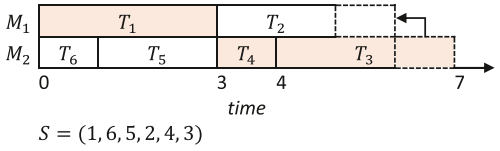


(b) Results of assigning all incoming tasks to a single machine.

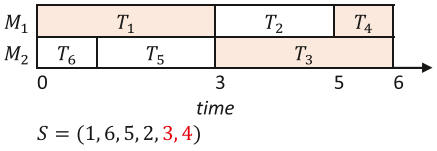


(c) Results of assigning all incoming tasks to a parallel machine.

Fig. 8. Illustration of how to allocate a set of submitted tasks to a range of available machines.



(a)



(b)

Fig. 9. Explanation of how to enhance the scheduling results shown in Fig. 8(c).

the root algorithms: heuristics. Heuristic algorithms depend on the nature of the problem and perform very well with certain problems while present low performance with others. Usually, an exact solution is provided by heuristics in an acceptable time with a specific kind of problem; however, they stumble in regard to hard optimization problems. In cloud environments, lots of heuristic algorithms have been proposed to solve the scheduling problems concerning workflow and independent tasks/applications. By observing the prime keywords in each selected article, some of the mainstream heuristic algorithms that have proven themselves in cloud computing are profoundly distinguishable, as per Table 5, into different categories, such as Min-Min [78], Max-Min [78], First Come First Serve (FCFS [79]), Heterogeneous Earliest Finish Time (HEFT [61]), Shortest Job First (SJF [80]), Round Robin (RR [81]), Minimum Completion Time (MCT [66]), and Sufferage [82]:

- **Min-Min** is a basic heuristic algorithm in which the task (shortest task) that will be executed within the minimal time is picked out of all the submitted tasks and mapped to a VM that will spend minimum completion time. This process continues until all tasks are successfully scheduled, thereby increasing the total makespan as the completion time of each individual task increases. This approach handles small tasks fluently, and large tasks have to wait until smaller ones are executed first. The overall system throughput is highly improved with this algorithm; however, the starvation problem might occur on heavy tasks.
- **Max-Min** is a bit like Min-Min, but in Max-Min, the longest task is chosen firstly and assigned to the most appropriate VM that should, among all the available VMs, execute the task within the minimum completion time. Compared to Min-Min, in Max-Min, the throughput is higher while makespan is less.
- **FCFS** assigns the incoming task based on its arrival time. It assigns the task whenever it arrives, to available resources, thus reducing the complexity and waiting time.
- **HEFT** is basically a list-based scheduling heuristic in which a task-priority list is firstly built so that optimal allocation decisions are then made locally for each task based on the task's estimated completion time.
- **SJF** is associated with the next CPU burst for every task's length. The task with the shortest length is assigned to a CPU whenever this CPU is available. IF there are two tasks with the same CPU burst time, then FCFS is incorporated for making a proper scheduling decision.
- **RR** triggers at the arrival time to immediately allocate available resources to the incoming task; however, the resources are provisioned to the task for a certain amount of time (time quantum). Then, if more execution time is still required, the task is preempted, queued, and awaiting for its execution to be resumed later.
- **MCT** schedules the tasks based on their expected minimum execution time. However, the task may not necessarily have its MCT on the same VM.
- **Sufferage** is a heuristic technique in which a resource is mapped immediately with a task which would likely suffer the most according to a "sufferage" threshold value which is associated with its expected completion time. In sufferage, each task's completion time on each resource is first computed. Second, the difference between two consecutive MCTs for each task is calculated and referred to as the suffrage value. Finally, a resource with the minimum execution time is provisioned to the task with the maximum suffrage value, all resources' execution times are updated, and the above steps are recurring until all the tasks are scheduled successfully. Despite its perfect performance in many cases, this strategy has a shortcoming if multiple tasks have the same sufferage value, where the first incoming task is simply selected and first executed without considering other tasks, which may arise a starvation problem.

Although heuristics are obsolete, they are still under consideration by many researchers. As shown in Table 5, heuristic algorithms have revealed significant performance based on diverse scheduling approaches addressed in the literature. Amongst them, it is worth noting that the SJF algorithm reveals outperformance in terms of independency [56,83,84] and cooperation [85–87]. Moreover, as per Table 5, it is remarkable that there is no preponderant heuristic approach over all selected studies as each of the approaches involved is adaptable and thus can excel in accordance to the nature of the scheduling problem addressed. Finally, the most notable is that there is no even one heuristic algorithm that has outperformed the involved meta-heuristics, which further emphasizes our felicitous choice of meta-heuristics to be the focus of attention in this study.

Less popular, evolving heuristic algorithms that have successfully been applied to task scheduling problems in cloud environments include Greedy Randomized Adaptive Search Procedure (GRASP [109,110]), Branch & Bound (BB [111]) algorithm, Load-Balanced Min-Max (LBMM [112]), etc.

Table 5
Summary of some promising heuristic approaches for task scheduling in cloud computing.

Research	Technique(s) applied	Compared against	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Min-Min						
Chen et al. [88]	User-Priority Aware Load Balancing Min-Min (PA-LBMM)	Min-Min and LBIMM	Independent	• Low makespan • High resource utilization	• Ignoring deadline • Slow task execution due to rescheduling	Simulation (MATLAB)
Amalarethinam and Kavitha [89]	Reschedule-based Enhanced Min-Min (REMM)	Min-Min and LBMM	Independent	• Low makespan • High resource utilization	• Ignoring priority and cost • Slow task execution due to rescheduling	Simulation (N/A)
Max-Min						
Mao et al. [90]	Elastic Cloud-aware Max-Min (ECMM)	Max-Min and RR	Independent	• Low makespan • Low response time • High resource utilization	• Potential resource overutilization/ underutilization	Simulation (CloudSim)
Karuppan et al. [91]	Priority-Based Max-Min (PBMM)	SJF and RR	Independent	• Low execution time • High resource utilization	• Not considering emerging dynamic tasks • High potential of over-/under-utilized resources	Simulation (CloudSim)
FCFS						
Saeed et al. [92]	Native FCFS	RR and Throttled	Independent	• Low response time • Low energy consumption	• Wretched supremacy	Simulation (CloudAnalyst)
HEFT						
Dubey et al. [93]	Modified HEFT	HEFT and CPOP [94]	Workflow scheduling	• Low overload • Low makespan	• Non-continuous monitoring of nodes • Load imbalance	Simulation (CloudSim)
Zhou et al. [95]	Fuzzy dominance-based HEFT (FDHEFT)	ϵ-Fuzzy PSO [96], NSPSO [97], SPEA2* [98], and MOHEFT [99]	Workflow scheduling	• Low makespan • Low CPU runtime • Low monetary cost	• Cost and time overheads due to unmanaged communication and storage	Simulation (jMetal [100]) and real environment
Tong et al. [101]	HEFT with *Q*-Learning (QLHEFT)	HEFT_D, HEFT_U, and CPOP [102]	Workflow scheduling	• Low makespan • High throughput • Suitable for heterogeneous environments	• Static task scheduling • Single objective (makespan)	Simulation (WorkflowSim)
SJF						
Alworafi et al. [83]	Improved SJF	SJF and FCFS	Independent	• Low makespan • Low response time	• Potential starvation • Potential load imbalance	Simulation (CloudSim)
Nazar et al. [56]	Modified SJF	RR and Throttled	Independent	• Low execution time • Low response time	• The same monetary cost • The same response time in clusters	Simulation (CloudAnalyst)
Seth and Singh [84]	Dynamic Heterogeneous SJF (DHSJF)	SJF and FCFS	Independent	• Low makespan • Low energy consumption	• Not considering dependent tasks • Potential starvation	Simulation (CloudSim)
RR						
Devi et al. [57]	Improved Weighted RR (IWRR)	Static RR and WRR	Independent	• Low response time • High resource utilization	• Poor ability to balance the workload	Simulation (CloudSim)
Prassanna and Venkataraman [103]	Threshold-based Multi-objective Memetic Optimized RR (T-MMORRS)	MGA [104] and DPRA [105]	Workflow scheduling	• High scheduling efficiency • Low makspan • Low energy consumption	• Imbalanced load	Simulation (CloudSim)
MCT						
Mehdi et al. [106]	datacentre Load and Power consumption (DLP)	Green algorithm [107] and RR	Independent	• Low power consumption • High task distribution ratio	• Potential starvation	Simulation (GreenCloud)
Sufferage						
Krishnaveni and Janita [108]	Completion Time-based Sufferage Algorithm (CTSA)	Min-Min, Enhanced Min-Min, and Sufferage	Independent	• Low makespan • High resource utilization	• Not considering cost	Simulation (CloudSim)
Krishnaveni and Prakash [82]	Execution Time-based Sufferage Algorithm (ETSA)	Min-Min, Enhanced Min-Min, and Sufferage	Independent	• Low makespan • High resource utilization • Balanced load	• Not considering cost, storage cost, and deadline	Simulation (CloudSim)
Hybrid						
Elmougy et al. [85]	SJF and RR with Dynamic Quantum (SRDQ)	SJF, RR, Time Slice Priority-Based RR (TSPBRR), SJF, and RR with Static Quantum (SRSQ)	Independent	• High throughput • Low response time	• Potential starvation • Load imbalance	Simulation (CloudSim)
Alworafi et al. [86]	Hybrid Shortest-Longest Job First (HSLJF)	SJF, LJF, and RR	Independent	• Low makespan • Low response time • High resource utilization • High throughput	• Not considering cost • Load imbalance	Simulation (CloudSim)
Caranto et al. [87]	User-priority SJF-RR (SJFRR)	SJF and RR	Independent	• Low waiting time • Low average turnaround time	• Low coping with context switching	N/A

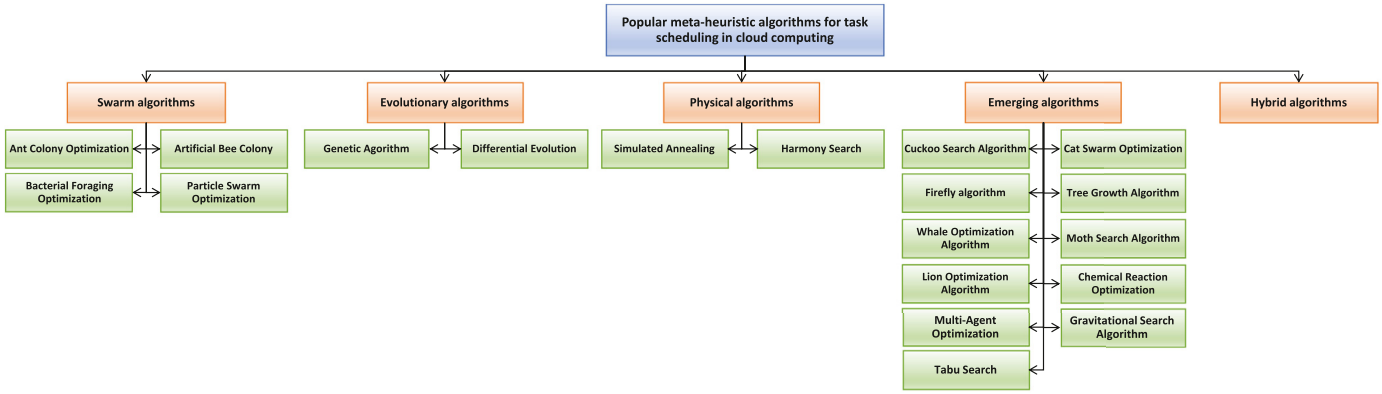


Fig. 10. Categorization of the meta-heuristic algorithms applied in cloud task scheduling.

4.3. Meta-heuristic scheduling

Over the past two decades, meta-heuristic algorithms have gained huge circulation because of their effectiveness in solving complex and large computational problems. Meta-heuristic algorithms have some useful features like: i) they are not problem dependent, ii) the search space is efficiently explored by these algorithms for finding a sub- or near-optimal solution to NP-complete problems, and iii) meta-heuristics are usually non-deterministic and based on approximation. Meta-heuristic algorithms are applicable and are esteemed for solving problems of different fields, with highly acceptable performance, by virtue of their instinctual independence of the problem to be solved [113,114]. Meta-heuristic strategies are commonly used as effective methods to solve NP-hard optimization problems, with high efficacy.

Meta-heuristic = Heuristic + Randomization.

Cloud environment is currently supported by various meta-heuristic algorithms which are adopted to obtain sub-optimal (approximate) solutions to NP-complete problems in a short time. Among many others, the process of task scheduling typically takes a long time for finding an optimal solution, especially with the large solution space. Thus, it is an appropriate fit for NP-complete problems.

This subsection describes a taxonomy of the mainstream meta-heuristic optimization algorithms applied successfully in the literature to solve task scheduling problems in cloud environments. The aims are to summarize and analyze the state-of-the-art, well-known meta-heuristics for solving task scheduling problems in cloud computing to follow the research trends [115]. In a related context, some emerging and hybrid meta-heuristic algorithms are also discussed. As shown in Fig. 10, diverse meta-heuristic scheduling techniques in the literature fall within one of three broad categories: swarm, evolutionary, and physical algorithms, based on the collective intelligence of large populations/swarms with simple behaviors for communication and interaction, the mechanism of biological evolution, and physical processes, respectively. In addition, two general categories (i.e., emerging algorithms and hybrid algorithms) are also involved to represent both novel and hybrid meta-heuristics, respectively. Generally, meta-heuristic algorithms are designed based on the mathematical models of various intelligent biological processes (e.g., genetic crossover, symbiosis, etc.) and activities (e.g., swarm foraging, bird flocking, etc.) occurring in nature, and adjusted and applied according to a given problem so that the resulting statistical data can be later interpreted, with respect to cloud service performance metrics, such as load balancing, resource utilization, power consumption, etc., to make optimal use of available cloud resources. For different swarm, evolutionary, physical, emerging, and hybrid algorithms, the frameworks and procedures are similar; however, the methods to initialize the population and evaluate the initial fitnesses of its agents, the strategies to generate new solutions, the iterative steps are usually different.

4.3.1. Swarm algorithms

Swarm intelligence algorithms [116], such as ACO [117], Artificial Bee Colony (ABC [118]), Bacterial Foraging Optimization (BFO [119]), and PSO [120], are developed based on mathematical models inspired by the activities as well as collective, cooperative behavior of different species, such as ants, honey bees, bacteria, and birds, respectively, which live in groups and collaborate for searching and gathering food. This being said, most conclusions in this manuscript implies the current large, rapidly growing bibliography of swarm intelligence. However, we concentrate our discussion on the-state-of-art meta-heuristics, such as ACO, ABC, BFO, and PSO, since they are arguably the first methods that fell within the swarm intelligence umbrella, and due to their relative maturity and higher prominence, especially with complex large-scale problems in the context of cloud computing [121].

Ant Colony Optimization (ACO): The natural behavior of ants in ACO helps significantly find the best traverse between the colonies and the food source [122]. In 1992, this approach was originally proposed as the “ant system” [117]. The ants evacuate the pheromones as they move on their path. When time goes by, pheromones shape the shortest pathways and the intensity of the pheromone allows to determine the shortest route to the food supply. Indeed, the ACO is inspired by the ants’ behavior of how to detect the shortest path between the anthill and the food source location [122]. In ACO-based cloud scheduling, the tasks are denoted by ants, and whole information about frequent use as well as the load on each resource can be indicated by the concentration of pheromones. Furthermore, the cloud resources (VMs) can be represented by food sources.

Artificial Bee Colony (ABC): The ABC algorithm was originally proposed by Lucic and Teodorovic [118]. This algorithm mimics the bees’ behavior on how to search and forage food. The ABC uses the effective concept to produce solutions from scratch in the execution step. The ABC has two alternatives, forward- and backward-pass. In forward-pass, every bee visits all solution components and reverts to the hive after making a partial solution [123]. Before the backward pass, the bees surround each other in a group, using their newly acquired minor individual solutions to start to retreat. In ABC-inspired cloud scheduling, the beehive describes the cloud environment, the tasks are illustrated by each artificial bee colony, while the VMs symbolize food sources. Bees searching food is the same as loading tasks to VMs, and looking for better sources of food can be imagined as finding the suitable low-load VMs to which incoming tasks are to be assigned.

Bacterial Foraging Optimization (BFO): The BFO was invented by Muller et al. [119] in 2002. The BFO is inspired by the collective behavior of bacteria (e.g., M.xanthus and E.coli), on how they forage. Specifically, the BFO is primarily based on the chemotaxis behavior of bacteria, in which they move towards or away from particular signals based on their ability to discern chemical gradients (e.g., nutrients) in the surrounding environment. This algorithm uses an information pro-

cessing strategy, where cells are allowed to collectively and stochastically swarm toward optima. In order to achieve this, a series of three processes is sequentially applied to a group of simulated cells [124]: i) 'chemotaxis', where cells are brought closer together to derate their cost, and cells move, one at a time, along the manipulated cost surface (the majority of algorithm work), ii) 'reproduction', where the upcoming generation can get benefits from only those cells that revealed high performance over their lifetime, and iii) 'elimination-dispersal', where new samples of cells are generated randomly and incorporated, with a low probability, for each cell after discarding old ones. In BFO-based cloud scheduling, the BFO is used to find an approximate solution to complex problems, where each cell can represent a cloud task mapping, the nutrients may express the resources (VMs), and the search space dimensionality can be represented by the number of cells (cloud tasks).

Particle Swarm Optimization (PSO): In 1995, Eberhart and Kennedy [120] employed the social behavior of particles to prompt the PSO algorithm. The PSO strives to solve an optimization problem by first generating a random set of potential solutions (particles). Then, each particle moves through the search space at its allocated elementary velocity. In each iteration, the velocity of each particle is regulated using both the best case of this particle and the case of the entire population's best solution (best particle). To balance the utilization, both local and global search methods are used in this technique. This optimization method has acquired a great popularity due to its simplicity and usefulness in working out a diverse set of applications at a low computational cost [125]. Accordingly, it has been also adapted to solve scheduling problems in clouds. In cloud scheduling techniques based on PSO, the number of tasks is the aspects of the solution (the particle) and each position can symbolize a set of candidate VMs similarly to the allocation process between incoming tasks and available VMs.

4.3.2. Evolutionary algorithms

Evolutionary algorithms (EAs), an algorithmic branch denoted as evolutionary computation [126], are typically used for large space problems whose sample space is not clearly defined. In EAs, the optimal solution is found by developing an incipient group of candidate solutions. Basically, the key mechanisms responsible for biological evolution (e.g., genetic crossover, selection, and mutation) are mathematically modelled to design EAs like Genetic Algorithms (GAs [127]). Like other meta-heuristics, EAs generate an optimal solution to a particular problem by following an iterative fashion.

Genetic Algorithm (GA): The GA, a population-based method of optimization, was first proposed in 1975 by Holland[128] and later upgraded by Khanli et al. [129] based on the evolutionary model inspired by nature. Looking closer at GA, every chromosome (individual in the population) contains a string of genes which substantially represents a potential solution. A fitness function is also provided in order to check whether the chromosome is suitable or not for the environment. The chromosomes are chosen on the basis of fitness value, and then the operations of crossover and mutation are performed to produce new offsprings (new potential solutions) which are used to compose the new population. The fitness function is then used to evaluate the quality of each offspring. This process is repeated until the production of an adequate umber of offsprings [130]. In cloud scheduling techniques based on the GA, the genes being swapped into the chromosomes to produce new offsprings can be considered as tasks to be assigned to VMs to run on them. The mapping sector simply includes the indices of the VMs on which tasks are to be run.

Differential Evolution (DE): The DE, a kind of optimization algorithm whose real vector coding is in continuous space, has the ability to perform a parallel and random search with the existence of simple, less-controlled parameters. This algorithm was firstly developed in 1995 by Storn and Price [131]. The basic idea of DE is similar to that of GA; that is, the mutation operation generates new individuals, then the crossover and selection operations are performed through a constant iterative evolution to find the global optimal solution. The DE implements the mutation operation using the difference strategy, which differs from the GA by enhancing the algorithm's ability to search based on the population characteristics. The DE algorithm proves significantly advantageous over many other algorithms in solving hard engineering problems including task scheduling in cloud environment. In cloud scheduling techniques based on the DE, D-dimensional parameter vectors may represent D tasks to be executed alternately and evaluated on a set of available VMs [132].

4.3.3. Physical algorithms

These methods are basically inspired by the laws of physics. The inspirational physics-based environments range from complex dynamic systems, like avalanches, to metallurgy, music, as well as the interplay between evolution and culture. These algorithms are generally stochastic with a mixture of global and local (neighborhood-based) search methods, which makes solvers inspired by them remarkable to tackle hard task scheduling optimization problems in cloud environments.

Simulated Annealing (SA): SA was originally introduced by Kirkpatrick [133] in 1984. Due to belonging to meta-heuristics, it is currently one of the most preferred algorithms. The SA process simulates the annealing process of materials, in which minimum energy and larger crystals are used to cool a metal crystallically so as to strengthen the metal structure. The annealing process involves two main aspects: precise temperature control and controlled cooling rate. SA was influentially practiced in diverse areas [134]. For example, many hard combinatorial optimization problems were worked out by the SA through a controlled random simulation of the way how temperature is dropped in certain thermodynamics systems. In cloud scheduling based on the SA, the number of tasks can represent the dimensions of the solution set, while the stable temperature may denote an efficient mapping reached between an incoming task and a suitable VM.

Harmony Search (HS): Geem et al. [135] implemented the HS as an evolutionary algorithm. In this algorithm, the cycle of musical improvisation is imitated, where each produced harmony is a possible solution. Initially, the harmony memory in HS stores a stochastic population of harmonies. Then, new solutions are implemented based on three principles: tone adjustment, random selection, and harmony memory consideration [136]. Due to being genetic, the HS falls under the vulnerary algorithms. The procedure of HS includes harmony memory, random selection, sound volume regulation, and bandwidth. The harmony memory level dictates whether new responses will be accepted or not. For more effective use of this memory, a parameter is used to help extract all possible selections from the harmony memory. The HS is also applied to the cloud scheduling problem, where each composed harmony can represent a potential task-VM allocation, and the competence level as well as the execution length of allocated tasks are highly based on how efficient is the transfer of submitted tasks to existing resources.

4.3.4. Emerging algorithms

In recent years, many novel meta-heuristic algorithms have been devised (apart from the discussed above) to be applied to different complex optimization problems in cloud computing, and they can be simply categorized among the three classes heading the three previous subsections. Here are some of those emerging algorithms: Cuckoo Search Algorithm (CSA [137]), Cat Swarm Optimization (CSO [138]) algorithm, Firefly Algorithm (FA [137]), Tree Growth Algorithm (TGA [139]), Whale Optimization Algorithm (WOA [140]), and Moth Search Algorithm (MSA [141]), which are considered as swarm algorithms; Lion Optimization Algorithm (LOA [142]), Chemical Reaction Optimization (CRO [143]), and Multi-Agent Optimization (MAO [144]) algorithm, which are considered as evolutionary algorithms; and Gravitational Search Algorithm (GSA [145]) and Tabu Search (TS [146]), which are considered as physical algorithms. Indeed, those state-of-the-art algorithms are specially addressed since they have shown significant success in the literature through the studies picked for this review.

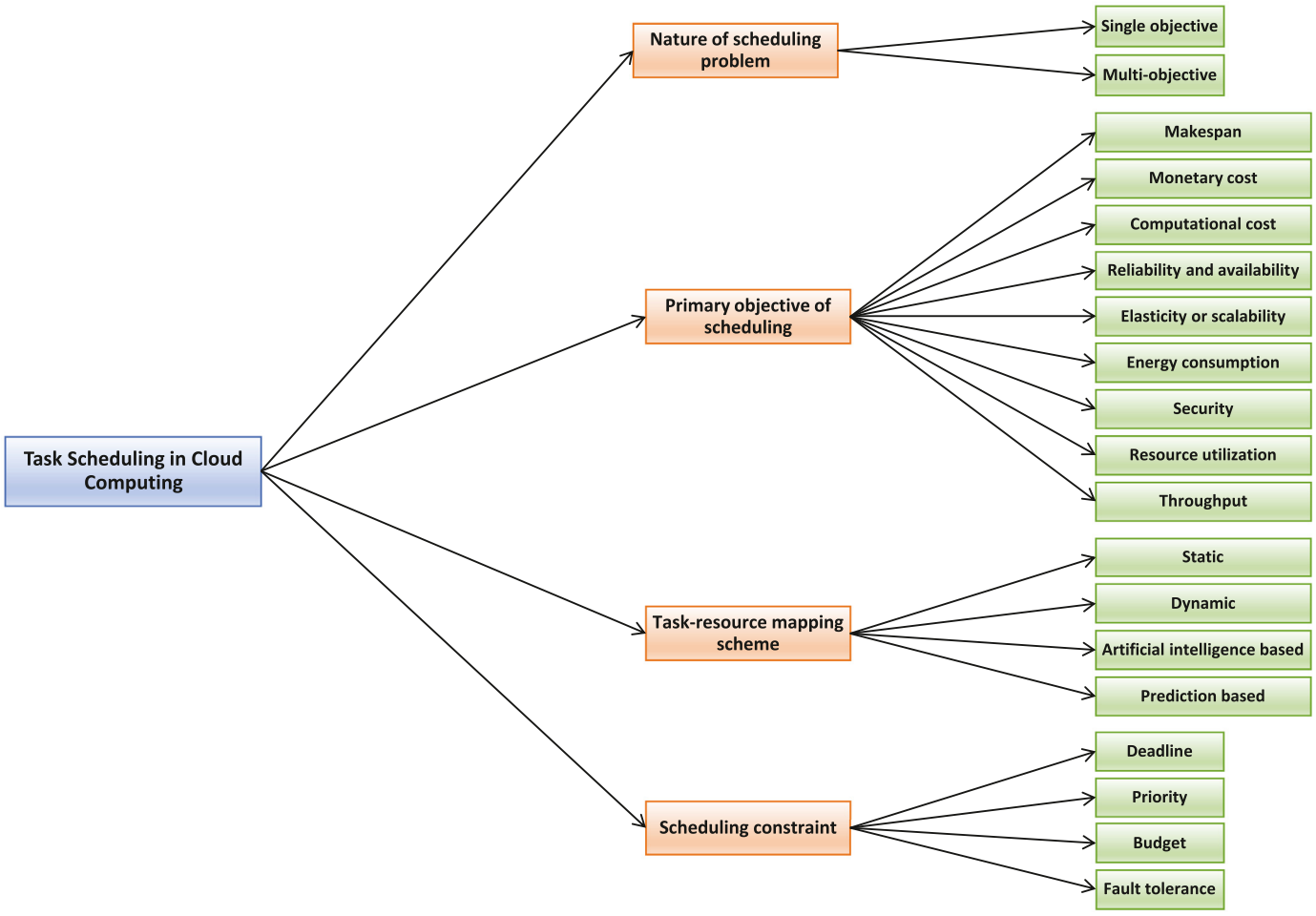


Fig. 11. Categorization of meta-heuristics based approaches in cloud task scheduling.

4.3.5. Hybrid algorithms

Two or more scheduling algorithms are combined in the hybrid scheduling algorithms in order to solve the task scheduling problem in the cloud environment. The underlying idea of hybrid algorithms is to leverage the strengths of diverse algorithms by hybridizing them into one algorithm in order to boost the performance in terms of either the computational time or the quality of the result or both. Hybrid scheduling algorithms include three different common kinds of combinations made by combining: i) single-solution based algorithm with population-based algorithm, ii) two population-based algorithms, or iii) a heuristic/meta-heuristic algorithm with another heuristic/meta-heuristic algorithm or technique. In this study, many different hybrid algorithms are discussed within the taxonomy introduced in Section 5. Moreover, a thorough analysis and discussion on those algorithms are put forward within the different aspects analyzed in Section 7.

5. Taxonomy of meta-heuristic task scheduling approaches in cloud computing

According to the QoS standard requirements, the aim of task scheduling varies from one application to another. Consequently, a large number of researches have been conducted in the context of meta-heuristics based task scheduling. To deeply and more clearly understand the meta-heuristic task scheduling approaches in cloud computing, a novel, rigorous taxonomy is provided as Fig. 11, using a variety of major methods followed in the literature. These methods are divided in this section into four main groups in terms of the nature of the scheduling problem, the primary objective of scheduling, the task-resource mapping scheme, as well as the scheduling constraint. The referred scheduling algorithms are also categorized into dependent tasks (realistic workflow scheduling) and independent tasks, based on the relation between the incoming tasks/applications. In addition, they are also labelled as traditional/heuristic or meta-heuristics (swarm, evolutionary, physical, or hybrid), based on the type of scheduling algorithm (scheduler). For performance evaluation, each technique evoked in every selected study is supposed to have been compared against many/multiple other peer algorithms, which are also outlined in the tables provided in what follows through in this section.

5.1. Nature of scheduling problem

Since there is always a trade-off between optimization objectives, there is a need to create an optimization model that fulfills the objectives by finding the best optimal solution. In a single objective optimization, it is possible to determine the optimality of a particular solution in comparison with another existing one. While in Multi-objective Optimization Problems (MOPs), it is not possible to do it directly. Furthermore, in single objective optimization problems, a single optimal solution is picked up for predefined objectives, while MOPs normally use a Pareto dominance relation technique [147] for creating a comparison model that is used to replace a single optimal solution with a range of alternatives, thus providing various, numerous trade-offs between the objectives. For performance evaluation, only one solution must be selected from many Pareto optimal solutions provided in MOPs. By spotting their key characteristics, an overview of the mechanisms in some

of the selected studies is presented in the rest of this section, based on single and multi-objective optimization methods.

5.1.1. Overview of the selected mechanisms

Single objective: Most existing methods in cloud computing consider only the CPU and memory requirements when it comes to task scheduling, without even addressing the makespan requirement. For getting better performance, Mandal and Acharyya [148] introduced a novel method using FA to schedule submitted tasks in clouds, aiming to reduce makespan. They prove the superiority of their algorithm by comparing the results with the CSA and SA. In this work, the authors focused on the single objective of minimizing the makespan time, ignoring other parameters, such as monetary cost, scalability, and availability.

With the same single objective of minimizing the makespan time to execute the applications submitted, Elaziz et al. [149] proposed a hybrid task scheduling algorithm by bringing two meta-heuristic algorithms, the MSA and DE algorithms, together as a hybrid algorithm, namely, MSDE. Using the concepts of Lévy flight and phototaxis, the exploration and exploitation capabilities are provided by MSA; however, the MSA exploitation capability is limited, so DE algorithm is used for local search to overcome the limitation of MSA by providing better exploitation capability. CloudSim toolkit is used in this study for three experimental series that demonstrate that the hybrid MSDE algorithm outpaces the state-of-the-art heuristic and meta-heuristic scheduling algorithms in terms of makespan and the throughput of the system. However, resource utilization, energy consumption, scalability, and availability are not considered.

Multi-objective: Task scheduling in a distributed heterogeneous computing environment can be identified as a non-linear, multi-objective, NP-hard optimization problem that strives to optimize cloud resource utilization and satisfy the QoS requirements. In this regard, Ramezani et al. [150] developed a comprehensive multi-objective model with conflicting objective functions to minimize task execution/transferring time and cost, depending on the speed and accuracy of the PSO algorithm. The proposed model is implemented and evaluated by extending the Jswarm package to multi-objective (MO-Jswarm) using the Cloudsim toolkit. The proposed optimization model achieves the best trade-off solution and the highest QoS in comparison with existing task scheduling approaches. However, energy consumption as well as task priority & type are not undertaken.

Once again, Ramezani et al. [151] extended the objectives to include task queue length and power consumption by proposing two algorithms: Multi-Objective PSO (MOPSO) and Multi-Objective GA (MOGA). In the proposed algorithms, the optimal task allocation among VMs is determined by extending the Cloudsim toolkit to apply the task scheduling algorithms of MOPSO and MOGA. The experimental results have shown that MOPSO has a higher efficiency and reliability on determining the best scheduling scheme with the highest QoS in the shortest possible time in terms of task queue length (VMs' workload), task execution cost, task transfer time, and power consumption. However, the priority of tasks and their types remains unresolved. Moreover, further criteria of SLA need to be covered.

In [152], Zuo et al. proposed a multi-objective ACO-based optimization method referred to as (PBACO) to achieve two objectives: optimizing both the scheduling Performance and the user's Budget. They consider the two constraints of makespan and the user's budget to prevent the ACO algorithm from stucking into the local optimum by evaluating the costs and providing feedback, in a timely manner, on the quality of the solution. Experimental results have shown a higher performance of this multi-objective optimization method than similar methods in terms of makespan, cost, deadline violation rate, and resource utilization. However, energy consumption, availability, and scalability are not minded.

Achieving a reasonable trade-off among energy consumption, resource utilization, and QoS requirements is a challenging problem, especially with diverse tasks in the heterogeneous environment. Therefore, He et al. [153] took into account both transmission time and processing time of tasks in a proposed strategy called Adaptive Multi-objective Task Scheduling (AMTS) based on the PSO. The PSO-based AMTS algorithm uses the Small Position Value (SPV) rule to convert its continuous position values to a discrete task permutation. Experimental results have shown a better quasi-optimal solution obtained by the PSO-based AMTS algorithm in terms of average cost, task completion time, and energy consumption. However, service availability and scalability are not assumed.

An innovative algorithm in cloud computing called Multi-Objective CSO (MOCSO) was introduced by Madni et al. for dealing with the problem of Infrastructure as a Service (IaaS)-related resource scheduling [154]. The proposed algorithm aims at reducing the cloud user cost and minimizing the makespan time, contributing to generating more profit/revenue to cloud providers as well as achieving maximum utilization of resources. Nevertheless, energy consumption, resource utilization, availability, and scalability are not analyzed in this work.

5.1.2. Summary of the scheduling approaches reviewed based on the nature of scheduling problem

Some of the selected articles have been studied and constructively commented in the previous subsection, along with their advantages and weaknesses, based on the nature of the scheduling problem focused in the study. A summary comparison between those articles, featuring their most important merits and compelling demerits, is illustrated in Table 6.

5.2. Primary objective of scheduling

For getting a high performance, at least one objective function is needed when a task scheduling process is performed. The most popular objectives can be summarized as: makespan, monetary cost, computational cost (i.e., consumption of CPU, memory, storage, GPU, bandwidth, etc.), reliability and availability, elasticity or scalability, energy consumption, security, resource utilization, and throughput. By highlighting their main features, a comparison between some of the selected articles is introduced in the rest of this section, based on the primary scheduling objective pursued in each study.

5.2.1. Overview of the selected mechanisms

Makespan: At cloud datacentres, inefficient task scheduling may reduce revenue as a result of resource underutilization. In this context, to perform efficient scheduling of tasks on cloud, the makespan needs to be reduced. In a study, Raju et al. [160] proposed a hybrid algorithm of ACO and CSA as a resource scheduling policy to reduce makespan. The hybrid algorithm might help reduce the makespan or completion time as it allocates the required resources optimally in such a way that the submitted jobs are executed on time. However, fault tolerance, resource utilization, and scalability are not undertaken.

For solving the optimization problem of workload scheduling in cloud computing, Khalili and Babamir [161] presented a single objective PSO algorithm. Different inertia weight strategies are combined with the PSO algorithm in order to minimize the makespan. The results have shown an improvement in the makespan obtained from the combination of Linearly Decreasing Inertia Weight (LDIW) with PSO. In the same context, Gabi et al. [162] introduced a conventional CSO task scheduling technique incorporated with an LDIW equation in order to overcome the entrapment problem arising from the local search in the technique. On CloudSim simulator tool, this technique minimizes the makespan time by performing an efficient task mapping to VMs based on enhanced convergence speed. However, all other QoS parameters than makespan are not touched.

In [163], Malik and Jain discussed an HS-based scheduling approach, aiming to fulfill the user requests in terms of the minimum execution time (i.e., minimum makespan) and efficient using of resources (i.e., resource utilization). However, the priority of tasks, energy consumption, availability, and scalability are not included in this work.

Table 6
Categorization of the techniques reviewed based on the nature of the scheduling problem, along with their advantages and weaknesses.

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Single objective						
Mandal and Acharyya [148]	FA, CSA, and SA *Meta-heuristic (swarm and physical)*	Each other *Meta-heuristic (swarm and physical)*	Independent	• Low makespan	• Low reliability	Simulation (GCC compiler)
Elaziz et al. [149]	Hybrid (MSA with DE) *Meta-heuristic (swarm and evolutionary)*	SJF, RR, MSA, WOA, and PSO *Heuristic and meta-heuristic (swarm)*	Independent	• Low makespan • High throughput	• High time complexity	Simulation (CloudSim)
Multi-objective						
Ramezani et al. [155]	MOPSO *Meta-heuristic (swarm)*	Three PSO variants and method in [156] *Meta-heuristic (swarm)*	Independent	• Low execution/transferring time • Low execution cost	• Low reliability • High energy consumption	Simulation (CloudSim)
Ramezani et al. [151]	MOPSO/MOGA-based algorithm *Meta-heuristic (swarm and evolutionary)*	Improved PSO [157–159] and each other *Meta-heuristic (swarm and evolutionary)*	Independent	• Low response time • Low makespan • High throughput • Low energy consumption	• Low reliability	Simulation (CloudSim)
Zuo et al. [152]	PBACO *Meta-heuristic (swarm)*	Min-Min, FCFS, and ACO *Heuristic and meta-heuristic (swarm)*	Independent and workflow scheduling	• Low makespan • Low monetary cost • High resource utilization • Minor deadline violation	• Low scalability • Low reliability	Simulation (CloudSim) and real environment
He et al. [153]	PSO *Meta-heuristic (swarm)*	GA *Meta-heuristic (evolutionary)*	Independent	• Low makespan • Low monetary cost • Low energy consumption • High resource utilization	• Low availability • Low scalability	Simulation (CloudSim)
Madni et al. [154]	CSA *Meta-heuristic (swarm)*	Min-Min, ACO, PSO, and GA *Heuristic and meta-heuristic (swarm and evolutionary)*	Workflow scheduling	• Low makespan • Low monetary cost • High resource utilization	• Low reliability • Low availability	Simulation (CloudSim)

On a modern cloud datacentre architecture, Sharma and Garg [164] tackled an energy-efficient task scheduling problem by proposing a novel hybrid Harmony-Inspired GA (HIGA) scheme for real-world scientific workflows. Without the need for many iterations, HIGA intelligently senses the local and global search space through combining the GA exploration capability and HS exploitation capability, thereby providing quick convergence. The objectives in this work are conceived for reducing the makespan, the computing energy, the energy consumed by the resources, and the execution overhead associated with a scheduler. However, only the independent tasks are considered in this study. Moreover, execution time estimation for real-time tasks is not incorporated.

Monetary cost: In [165], Meena et al. proposed a Cost-Effective GA (CEGA); that is, while striving to meet the deadline, the workflow execution cost decreases in the cloud computing environment. New schemes of crossover, mutation, population initialization, and encoding operators of GA are developed to achieve the desired goal. In the case of a small number of tasks, CEGA gives a good solution, whereas in high complexity applications, it consumes a higher makespan.

An efficient load scheduling aims to optimize the load between VMs and tasks in cloud computing. Therefore, a load scheduling approach based on New PSO (NPSO) was proposed by Chaudhary et al. [166], in which a new cost evaluation function is used to minimize the monetary cost needed for processing the tasks on VMs. However, this approach lacks realism in application and a sufficient comparison with a reasonable number of existing techniques.

Han et al. [167] proposed an algorithm, namely, HDEA based on a prevalent DE meta-Heuristic algorithm as well as several optimization policies to optimize task scheduling in the cloud environment in terms of monetary cost and turnaround time, depending on two methods for generating the initial population by adopting a parameter adjustment strategy and building a new mutation strategy and several local search methods to obtain better solutions. However, the scheduling of dependent tasks is not addressed in this work.

Nasr et al. [168] combined the CRO and ACO algorithms in a single hybrid algorithm, namely, CR-AC, for solving the workflow scheduling problem under a deadline constraint with the purpose to reduce monetary cost and time complexity. However, important factors, such as energy consumption, fault tolerance, and security, are not examined in this study.

Distributed Green Clouds (DGCs) are adopted by more and more large-scale enterprises in recent years for managing their core business applications effectively in terms of monetary cost. However, this requires maximization of DGCs' profit, given the spatial differences in revenues, prices of power grid and Internet service provider bandwidth, and renewable energy. In this regard, Yuan and Bi [169] designed a profit maximization problem as a constrained non-linear optimization program and managed to tackle it using a hybrid meta-heuristic algorithm (PSO with GA and SA). In this way, the profit is optimized based on a Profit-Aware Spatial Task Scheduling (PASTS) method by scheduling all tasks smartly within their response time limits. Throughput is optimized in this scheme, as well. However, availability, energy consumption, and fault tolerance are not examined.

Computational cost (CPU, memory, storage, GPU, bandwidth, etc.): The total execution time can be minimized by provisioning a large amount of resources, but this might cause scheduling overheads, resource underutilization, and execution cost to increase. Therefore, Wu et al. [170] presented a market-oriented resource scheduling technique based on ACO, PSO, and GA, considering the task-level and service-level dynamic resource scheduling, where a task is assigned to a VM, and a

task is assigned to a service, respectively. This approach optimizes the makespan and CPU time as well as reduces the overall running cost of datacentres; however, allocation of resources to global tasks does not perform optimally in this method. In addition, availability, throughput, energy consumption, and fault tolerance are not inspected.

In [171], Thaman and Singh devised a new hybridization of a novel heuristic Nearest Neighbor (NN) with a variant of the meta-heuristic PSO, namely, Nearest Neighbor Cost-Aware PSO (NNCA_PSO). Allocation of tasks to resources is achieved by the heuristic NN through a reduction in variance between the execution time characteristics of resources and the completion time requirements of tasks. The NN scheduling algorithm, as alone, proves highly effective over other traditional heuristics for a set of independent tasks. The proposed NNCA_PSO also reveals higher performance than other cost-aware PSO variants. This research work addresses multiple related cloud issues to execution cost, makespan, resource utilization, and energy consumption. However, availability, scalability, and fault tolerance are not tackled.

The swarm intelligence-based load scheduling is stochastic, self-collective, decentralized, random, collective, intelligent, adaptive, and more dependent on bio-inspired mechanisms than conventional mechanisms. Proceeding from this fact, Chaudhary and Kumar [172] proposed a Hybrid Genetic-GSA (HG-GSA), a new load scheduling technique, for reducing the total cost of computation, including both execution cost and transfer cost through achieving the maximum resource utilization. However, this method does not consider makespan, energy consumption, availability, scalability, and fault tolerance.

Reliability and availability: Increasingly, applications are regularly deployed to the cloud environment due to its increasing popularity and high reliability. Among such applications, the most common example is web 2.0 applications. These applications require to be highly available and able to run for longer periods of time uninterruptedly (or even indefinitely). Therefore, a multi-objective scheduling algorithm was proposed by Frincu and Craciun [173] to achieve the mapping of application component instances to cloud resources in an efficient way through simultaneously optimizing three goals: i) resource usage is maximized, ii) application runtime cost is minimized through the optimal mapping of the component to a node by continually rescheduling of tasks if the node is still overloaded or underloaded, and iii) application availability is maximized by the even spreading of component instances across the allocated nodes. This algorithm is designed for efficiently deploying applications through picking out the best among public cloud providers or selecting a combination of them. Thus, hybrid cloud is not the best environment for this algorithm. Furthermore, energy consumption, scalability, monetary cost, and throughput are not contemplated.

In another work, Faragardi et al. solved the task scheduling problem using an SA algorithm hybridized with TS, hence making improvements in the native algorithms: In 2012, by using a non-monotonic cooling schedule [174]; and in 2013, by supplying the algorithm with more systematic memory to prevent cycling by storing the recently visited solutions [175]. In this method, the system reliability is maximized, but within a high execution time. Additionally, energy consumption, scalability, monetary cost, and throughput are not undertaken.

For conducting a reliability examination on cloud services, Cui et al. [176] managed to use the Markov model and the Queuing theory to present the scheduling problem of tasks in the cloud. Various objects, such as flow time, makespan, and reliability, are taken as targets of the optimization problem. The GA-based Chaotic Ant Swarm (GA-CAS) algorithm is proposed in this research for scheduling the tasks. An improved convergence rate is achieved by the proposed algorithm in terms of makespan, flowtime, and reliability. However, energy consumption, scalability, monetary cost, and throughput are not regarded.

Elasticity or scalability: In [177], a Cloud Scalable Multi-objective CSO-based SA (CSM-CSOSA) algorithm was carried out by Gabi et al., on one parallel workload and one dataset to fulfill the QoS requirements as well as the optimization problem constraints. For extending this study, in another study by Gabi et al. [178], a task scheduling model was built, solved on CloudSim framework using the CSM-CSOSA, and finally evaluated in terms of scalability, execution cost, and execution time. However, availability and energy consumption constraints are not valued.

For enhancing the scheduling process, Pradeep and Jacob [179] combined CSA with HS effectively into an CHSA to improve the optimization process by formulating a new multi-objective function for minimizing memory usage, energy consumption, cost, and penalty while maximizing credit. Herein, the CSA performance is improved by the HS algorithm for optimizing the task execution on VMs. However, throughput, fault tolerance, scalability, and availability are not addressed int this study.

Cloud computing enables a dynamic and efficient provision of distributed, scalable, and elastic resources to end-users from a finite pool of physical and virtual resources through scheduling of loads, based on enhanced versions of TGA. For example, an approach named dynamic search TGA (dynsTGA) was introduced for reducing the total cost and transfer time [180]. However, execution time, energy consumption, and monetary cost are not minded.

Energy consumption: The parameters of deadline and budget were mainly identified as objectives in most of the traditional scheduling algorithms; however, researchers have recently taken the energy consumption seriously as a contribution to expanding the green cloud space. In datacentres, many efficient technologies including Dynamic Voltage and Frequency Scaling/Dynamic Voltage Scaling (DVFS/DVS) technology (a neoteric advance in processor's design), resource hibernation, and memory optimization were utilized for reducing the energy consumption of processors and makespan using a bi-objective hybrid GA [181]. Furthermore, authors in [182] proposed a hybrid workflow scheduling algorithm; namely, Multi-Objective Discrete PSO based on DVFS (DVFS-MODPSO). In this work, performance metrics, such as makespan, cost, and energy consumption, are optimized in a discrete space and a set of non-dominated Pareto optimal solutions are produced. However, scalability, security, total cost, and availability are not deemed a priority in this study.

Recently, a new multi-objective task scheduling approach was devised based on some Pareto-based methods. Amongst them, Tao et al. [183] proposed a hybrid GA based on Case Library and Pareto Solution (CLPS) to implement solutions to the optimization problem of reducing both the makespan and energy consumption. However, this approach does not contemplate scalability, monetary cost, and availability.

To reduce energy consumption, Meshkati et al. [184] presented a Hybrid ABC and PSO Scheduling Framework (HSF) for reducing active nodes by turning off the unused nodes in addition to managing the VMs placement on physical nodes. Furthermore, in [185], Goyal and Chahal proposed an FA-inspired method to perform cloud load balancing. In this research, the brightness function expresses the load function. Less bright firefly is assumed to mean, have a lighter load; and the firefly with more light is assumed to mean the load is high. Each of the virtual nodes are assigned a threshold value that should not be exceeded by the node when it is under processing. The results have demonstrated improvement in throughput, minimization in energy consumption, and maximization in response time. However, this approach considers only independent tasks. In addition, scalability, fault tolerance, and reliability are not accepted in the proposed system.

Security: In [186], Abdulhamid et al. proposed a Global League Championship Algorithm (GBLCA) technique to globally schedule scientific applications in the cloud. A remarkable development rate between 14.44% and 46.41% has been shown by this algorithm on the makespan. Moreover, the time it takes (i.e., response time) for securely scheduling the applications is reduced. However, total execution cost, energy consumption, scalability, and fault tolerance are not put forward.

A security-cost-aware scheduling approach was proposed in [187] based on a PSO technique which addresses the constraints of risk rate and deadline so as to minimize the monetary cost. Nevertheless, energy consumption, scalability, makespan, and fault tolerance are not discussed.

In a study, Wen et al. [188] proposed a GA-based privacy-aware multi-objective workflow scheduling algorithm in which a range of Pareto trade-off solutions are provided to cloud customers, including both execution time and monetary cost. However, energy consumption, resource utilization, scalability, and fault tolerance are not valued.

Sujana et al. [189] presented Smart PSO (SPSO) and Smart Variable Neighborhood PSO (SVNPSO) algorithms to find an optimized schedule for achieving a trade-off between the security and the minimum possible cost and makespan to execute scientific workflows in the cloud environment, thereby achieving maximal resource utilization. However, energy consumption, scalability, and fault tolerance are not validated.

Recently, Thanka et al. [190] proposed an Improved Efficient ABC (IE-ABC) algorithm for meeting the security and QoS targets in the cloud environment. The ABC algorithm is modified for proving both a security-aware scheduling and an efficient service while the task is mapped to the most suitable VM, based on the QoS policies and the critical security level of the user. In each datacentre, a hive table is maintained for reducing the security risk, task migration, makespan, cost, and VMs' load balancing. However, fault tolerance, scalability, resource utilization, and energy consumption are not minded in this work.

Resource utilization: For more reliable resource allocation in clouds, Javanmardi et al. [191] proposed a hybrid model based on the GA as well as fuzzy theory. Two chromosomes for each incoming task/application are generated. Both chromosomes include the job length, but additionally, the first chromosome includes CPU speed whereas the second includes the bandwidth of resources. To override traditional crossover operator of an offspring, the fuzzy theory is used with these chromosomes. However, how the fuzzy theory is exploited for reducing execution time and cost is unclear. In addition, energy consumption, availability, security, and scalability are not acceptable.

In a study, Kumari and Jain [192] proposed a new approach combined of PSO with a Max-Min strategy to overcome the PSO limitation of taking a longer time for conversion to the global optimal solution. This approach is mainly compromised for minimizing the makespan and maximizing the resource utilization in the optimization problem of task scheduling. However, energy consumption, security, availability, scalability, and fault tolerance are not regarded.

In another research work, Rani and Suri [193] incorporated the GSA and ACO into a single hybrid task scheduling algorithm to solve the load balancing problem. Thanks to this combination, the ACO no longer has early convergence. In this paper, VMs are considered as potential food sources, and the concept of pheromone is exploited for finding the best VM. The GSA is utilized for updating the pheromone. The results reveal a balanced distribution of load achieved through relieving the overloading on VMs. This algorithm achieves load balancing by distributing the load fairly across all VMs as well as improving the task completion time, thereby reducing resource consumption. However, energy consumption, security, scalability, and fault tolerance are not evoked.

Finally, Chen et al. [194] applied a recently introduced meta-heuristic, WOA, for cloud task scheduling using a multi-objective optimization model. An advanced approach called Improved WOA for Cloud (IWC) is proposed to further improve the search capability of WOA. The proposed IWC achieves better accuracy and convergence speed, compared to existing meta-heuristic algorithms, in searching the optimal scheduling plans and improving the efficiency of the cloud system in terms of both system load and resource utilization for both small- and large-scale tasks. However, exploration-exploitation balance in IWC may be further improved. Workflow scheduling is not considered, as well.

Throughput: To get the maximal throughput, instead of migrating the entire overloaded VM, a novel model called Task-based System Load Balancing (TBSLB) was validated by Ramezani et al. using the PSO as a scheduler algorithm in order to migrate the running tasks from an overloaded VM to another homogeneous one, thereby achieving an optimal cloud load balancing. This algorithm includes also a model that optimizes task migration in order to reduce both the transfer and execution times of a task. The results have shown less time taken by this load balancing technique than traditional methods. In this method, the idle PMs are not selected such that the energy consumption is kept low. Hence, this method boosts the load balancing, decrease energy consumption, and reduces the execution and migration times, thereby increasing the throughput. However, only the homogeneous environment and independent tasks are considered. Furthermore, security, availability, and fault tolerance are not touched.

For estimating the load on VMs, Shobana et al. proposed an ABC-based preemptive task scheduling technique using the bandwidth and processor as decision metrics [195]. In this method, the scout bees collect the information while searching a food source, and if suitable nectar source is found, information is sent to onlooker bees. The tasks are executed immediately after the task scheduling process is exclusively and completely performed. Indeed, the bees' information including task priority, number of tasks, and location of VMs is updated by the onlooker bees on an appropriate set of VMs. It is assumed that the summation of all VMs' loads on a PM matches the entire load on each of the other PMs as much as possible. The proposed algorithm imitates the honeybees' foraging behavior for minimizing the makespan. Also, the priorities of tasks are examined and exploited in this algorithm with the aim of reducing latency and increasing overall efficiency. The proposed method utilizes resources efficiently for improving the response time for end-users. However, the proposed technique considers only independent tasks. In addition, low reliability is presented, as well as energy consumption, security, availability, and fault tolerance are not included.

Finally, a new resource scheduling technique based on a Gradient Descent (GD) approach hybridized with the CSA was proposed by Madni et al. [196] for resolving and optimizing the resource scheduling in clouds. All entities in the CSA are searching the same way. However, this standard search activity is not always helpful in finding a good solution to a particular problem, especially with the existence of the dilemmas of local optima and premature convergence. To overcome this weakness, the CSA is improved using the GD approach by enhancing the convergence rate, aiming to allocate an incoming task to a specific VM that ensures the lowest execution time, thereby increasing resource utilization for cloud providers while fulfilling the cloud users' demand with less delay. Evaluating the performance against existing meta-heuristic approaches as well as load balancing, makespan, and throughput are verified in this approach. However, the performance metrics of resource utilization and energy consumption are not investigated.

5.2.2. Summary of the scheduling approaches reviewed based on the primary scheduling objective

Some of the selected articles have been studied and constructively commented in the previous subsection, along with their advantages and weaknesses, based on the primary scheduling objective in the study. A summary comparison between those articles, including their remarkable merits and compulsory demerits, is illustrated in Table 7.

5.3. Task-resource mapping scheme

Static, dynamic, AI-based, and prediction-based mapping of cloud resources to incoming tasks is performed in order to efficiently use the available resources based on the cloud environment condition and the submitted workload. As is well-known, resources and workloads carry uncertainty in their characteristics; in addition, they are moldable in nature. Thus, the above-mentioned allocation schemes are developed and incorporated to handle QoS requirements and mitigate SLA violations. The most popular mapping schemes, along with their features, are discussed in the following subsection.

5.3.1. Overview of the selected mechanisms

Static: Static scheduling requires prior information regarding the tasks for making a schedule decision before a task starts to execute. Babu and Krishna [214] proposed a resource provisioning

Table 7
Categorization of the techniques reviewed based on the primary scheduling objective, along with their advantages and weaknesses.

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Makespan						
Raju et al. [160]	Hybrid (ACO and CSA) *Meta-heuristic (swarm)*	ACO *Meta-heuristic (swarm)*	Independent	• Low makespan • High throughput	• Low energy consumption • Low reliability • Low resource utilization	Real environment
Khalili and Babamir [161]	PSO with LDIW *Meta-heuristic (swarm)*	FCFS *Heuristic*	Independent	• Low makespan • High resource utilization	• Low reliability • Low throughput	Simulation (CloudSim)
Gabi et al. [162]	CSO with LDIW *Meta-heuristic (swarm)*	CSO & PSO with LDIW *Meta-heuristic (swarm)*	Independent	• Low makespan • High resource utilization	• Low scalability • High computational cost	Simulation (CloudSim)
Malik and Jain [163]	HS *Meta-heuristic (physical)*	N/A *N/A*	Independent	• Low makespan	• Low reliability • Low scalability • High energy consumption	Simulation (CloudSim)
Sharma and Garg [164]	Hybrid (GA with HS) *Meta-heuristic (physical and evolutionary)*	Min-Min, Max-Min, HEFT, PSO, HS, and GA *Heuristic and meta-heuristic (swarm, physical, and evolutionary)*	Independent and workflow scheduling	• Low makespan • Low energy consumption	• Low reliability • Low availability	Simulation (CloudSim with MATLAB)
Monetary cost						
Meena et al. [165]	GA *Meta-heuristic (evolutionary)*	IC-PCP [197], RCT [198], RTC [198], and PSO [199,200] *Heuristic and meta-heuristic (swarm)*	Workflow scheduling	• Low monetary cost • Minor deadline violation	• Low scalability	Real environment
Chaudhary et al. [166]	NPSO *Meta-heuristic (swarm)*	PSO *Meta-heuristic (swarm)*	Independent	• Low monetary cost	• Low reliability	Simulation (CloudSim)
Han et al. [167]	HDEA *Meta-heuristic (evolutionary)*	DE and SOS [201] *Meta-heuristic (evolutionary)*	Independent	• Low monetary cost • Low makespan	• Low reliability	Simulation (CloudSim)
Nasr et al. [168]	Hybrid (ACO with CRO) *Meta-heuristic (swarm and evolutionary)*	ACO, modified PSO, CRO, and CEGA *Meta-heuristic (swarm and evolutionary)*	Workflow scheduling	• Low monetary cost • Low makespan • Minor deadline violation	• Low reliability	Simulation (CloudSim)
Yuan and Bi [169]	Hybrid (PSO with GA and SA) *Meta-heuristic (swarm, evolutionary, and physical)*	EcoPower [202] and TRS [203] *N/A*	Independent	• Low monetary cost • High throughput • Low energy consumption • Low bandwidth	• Low reliability • High makespan	Real environment
Computational cost (CPU, memory, storage, GPU, bandwidth, etc.)						
Wu et al. [170]	ACO, PSO, and GA *Meta-heuristic (swarm and evolutionary)*	Each other *Meta-heuristic (swarm and evolutionary)*	Workflow scheduling	• Low computational cost • Low makespan	• Low scalability • Low reliability	Real environment
Thaman and Singh [171]	Hybrid (PSO with NN) *Meta-heuristic (swarm) and ML*	PSO *Meta-heuristic (swarm)*	Independent	• Low computational cost • Low monetary cost • Low makespan • Low response time • High resource utilization • Low energy consumption	• Low reliability • Low scalability	Simulation (MATLAB)
Chaudhary and Kumar [172]	Hybrid (GSA and GA) *Meta-heuristic (physical and evolutionary)*	PSO, LIGSA-C, and Cloudy-GSA [204] *Meta-heuristic (swarm and physical)*	Independent	• Low computational cost • High resource utilization • Minor SLA violation	• High total cost • Low scalability	Simulation (CloudSim)
Reliability and availability						
Frincu and Craciun [173]	GA *Meta-heuristic (evolutionary)*	RR *Heuristic*	Independent	• High availability • Low computational cost • High resource utilization • High fault tolerance	• Low scalability • Low reliability • High complexity	Real environment

(*continued on next page*)

Table 7 (*continued*)

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Faragardi et al. [174,175]	Hybrid (SA with TS) *Meta-heuristic (physical)*	Branch & Bound, SA, and GA *Heuristic and meta-heuristic (physical and evolutionary)*	Independent	• High reliability • High scalability	• High execution time	Simulation (C+)
Cui et al. [176]	Hybrid (chaotic ACO and GA) *Meta-heuristic (swarm and evolutionary)*	ACO, GA, and GSA (GA & SA) *Meta-heuristic (swarm, evolutionary, and physical)*	Independent	• High reliability • Low makespan • High fault tolerance • High resource utilization	• High complexity • Unbalanced load	Simulation (MATLAB)
Elasticity or scalability						
Gabi et al. [177,178]	Hybrid (SA with CSO) *Meta-heuristic (physical and swarm)*	GA, ACO, and PSO *Meta-heuristic (physical and swarm)*	Workflow scheduling	• High scalability • Low makespan • Low monetary cost • High throughput	• Low reliability	Simulation (CloudSim)
Pradeep and Jacob [179]	Hybrid (CSA with HS) *Meta-heuristic (swarm and physical)*	CSA, (CSA with GSA), and HS *Meta-heuristic (swarm and physical)*	Independent	• High scalability • Low makespan • Low computational cost • Low monetary cost • Low energy consumption	• Low throughput	Simulation (CloudSim)
Strumberger et al. [180]	dynsTGA *Meta-heuristic (swarm)*	Min-Min, FCFS, PSO, GA, SA, Cloudy-GSA, and TS *Heuristic and meta-heuristic (swarm, evolutionary, and physical)*	Independent	• High scalability • Low transfer time • Low computational cost	• Low reliability • Low throughput	Simulation (CloudSim)
Energy consumption						
Mezmaz et al. [181]	Hybrid (ECS [205] with GA) *Heuristic and meta-heuristic (evolutionary)*	ECS [205] *Heuristic*	Independent	• Low energy consumption • Low makespan	• Low scalability • Low reliability	Real environment
Yassa et al. [182]	PSO *Meta-heuristic (swarm)*	HEFT *Heuristic*	Workflow scheduling	• Low energy consumption • Low makespan • Low monetary cost	• Low reliability • Low security	Simulation (N/A)
Tao et al. [183]	Hybrid (GA with CLPS) *Meta-heuristic (evolutionary)*	Existing enhanced GAs *Meta-heuristic (evolutionary)*	Independent	• Low energy consumption • Low makespan • High stability	• Low reliability • Low scalability	Simulation (MATLAB)
Goyal and Chahal [185]	FA *Meta-heuristic (swarm)*	PSO *Meta-heuristic (swarm)*	Independent	• Low energy consumption • Low response time • High throughput	• Low reliability	Simulation (CloudSim)
Meshkati et al. [184]	Hybrid (ABC and PSO) *Meta-heuristic (swarm)*	FFD [206,207], ABC, and PSO *Heuristic and meta-heuristic (swarm)*	Independent	• Low energy consumption • Low makespan • High throughput • Minor SLA violation	• Low scalability • Low reliability	Simulation (CloudSim)
Security						
Abdulhamid et al. [186]	GBLCA *Meta-heuristic (evolutionary)*	Min-Min, Max-Min, ACO, and GA *Heuristic and meta-heuristic (swarm and evolutionary)*	Independent	• High security • Low makespan • Low response time	• Low scalability	Simulation (CloudSim)
Li et al. [187]	PSO *Meta-heuristic (swarm)*	N/A *N/A*	Workflow scheduling	• High security • Low monetary cost • Minor deadline violation	• Low scalability • Low reliability	Simulation (CloudSim) and real environment
Wen et al. [188]	GA *Meta-heuristic (evolutionary)*	MOPSO [208] and NSGA-II [209] *Meta-heuristic (swarm and evolutionary)*	Workflow scheduling	• High security • Low makespan • Low monetary cost	• High energy consumption	Simulation (CloudSim)
Sujana et al. [189]	SPSO and SVNPSO *Meta-heuristic (swarm)*	HEFT [102], PEFT [210], PSO, and RDPSO [211] *Heuristic and meta-heuristic (swarm)*	Workflow scheduling	• High security • Low monetary cost • Low computational cost • Low makespan • High resource utilization	• High complexity • Low reliability	Simulation (WorkflowSim)

(*continued on next page*)

Table 7 (*continued*)

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Thanka et al. [190]	IE-ABC *Meta-heuristic (swarm)*	ABC, ACO, and GA *Meta-heuristic (swarm and evolutionary)*	Independent and workflow scheduling	• High security • Low makespan • Low monetary cost • Low migration time	• Low scalability • Low reliability	Simulation (CloudSim)
Resource utilization						
Javanmardi et al. [191]	Hybrid (GA with fuzzy) *Meta-heuristic (evolutionary)*	ACO and MACO algorithms [212] *Meta-heuristic (swarm)*	Independent	• High resource utilization • Low makespan • Low computational cost	• Low reliability • High complexity	Simulation (CloudSim)
Kumari and Jain [192]	Hybrid (PSO and Max-Min) *Meta-heuristic (swarm) and heuristic*	Hybrid PBCOPSO (ABC and ACO) [213] *Meta-heuristic (swarm)*	Independent	• High resource utilization • Low makespan	• High energy consumption • High monetary cost	Simulation (CloudSim)
Rani and Suri [193]	Hybrid (ACO with GSA) *Meta-heuristic (swarm and physical)*	ACO and GSA *Meta-heuristic (swarm and physical)*	Independent	• High resource utilization • Low makespan • High throughput	• High monetary cost • Low reliability	Simulation (CloudSim)
Chen et al. [194]	IWC *Meta-heuristic (swarm)*	WOA, PSO, and ACO *Meta-heuristic (swarm)*	Independent	• High resource utilization • Low computational cost • Low monetary cost	• High scheduling overhead • Not considering workflows	Simulation (MATLAB)
Throughput						
Ramezani et al. [150]	PSO *Meta-heuristic (swarm)*	Traditional methods *N/A*	Independent	• High throughput • Low makespan • Low energy	• Low scalability • Low reliability	Simulation (CloudSim)
Shobana et al. [195]	ABC *Meta-heuristic (swarm)*	N/A *N/A*	Independent	• High throughput • Low makespan • Low response time	• Low reliability • Low scalability	Simulation (CloudSim)
Madni et al. [196]	Hybrid (CSA with GD) *Meta-heuristic (swarm) and heuristic*	ABC, ACO, PSO, CSA, LCA, GA, and SA *Meta-heuristic (swarm, evolutionary, and physical)*	Workflow scheduling	• High throughput • Low makespan	• High complexity • Low security • High energy consumption	Simulation (CloudSim)

mechanism inspired by the behavior of the ABC algorithm to improve resource utilization, increase the system throughput, and reduce queuing time by improving the load balancing among VMs and balancing the tasks' priorities on those VMs. For a task in the queue, the queuing time and execution time are lesser; however, the proposed approach does not validate the dependent tasks. Additionally, energy consumption, security, and availability are not scrutinized.

Matos et al. [215] proposed a solution based on a combination of: meta-heuristic GA, to minimize the number of required VMs by 20% and reduce the task processing time (makespan); and a static algorithm called Maximum Resource (MR), to resolve the set partitioning problem. However, the proposed algorithm presents a small limitation when the size of tasks increases by about 30%. In addition, energy consumption, monetary cost, reliability, and fault tolerance are not regarded.

Dynamic: Dynamic scheduling can occur during task execution and does not require knowledge of all task properties. This is useful for handling the fluctuating demands of cloud users, especially when maximizing the resource utilization is a higher priority than reducing the execution time [216].

Indeed, all meta-heuristic scheduling approaches are dynamic in nature [217]. However, in this subsection, some researches are addressed based on both the dynamic cloud environment and/or the dynamic scheduling scheme mentioned in the prime keyword of the selected articles. ACO-based and Variable Neighborhood PSO (VNPSO)-based dynamic scheduling algorithms were proposed by Islam and Habiba [218]. Several mathematical models and explanations of meta-heuristics are introduced to analyze an effective security-aware scheduling strategy. Meanwhile, the empirical results demonstrate the overall superiority of the proposed algorithm over other existing algorithms in terms of five basic metrics: security constraints, throughput optimization rate, monetary cost, and CPU utilization. However, energy consumption, monetary cost, and reliability are not considered.

In [219], a dynamic workflow scheduling technique was proposed by Rahman et al. in grid and cloud computing environment for reducing the scheduling overhead and minimizing the workflow execution time. In this approach, a task-to-service mapping scheme is built using the GA algorithm combined with a Dynamic Critical Path (DCP) algorithm, in which the task with the longest execution path (critical path) is dynamically determined posterior to scheduling the incoming tasks. However, this approach does not consider using the GA to obtain a better solution while workflow execution is running. In addition, no experimental research is presented for the suggested technique. In addition, no evidence is provided on the algorithm's ability to handle incoming workflows during runtime. However, in normal, GA is given a while to make a better schedule for the newly incoming workflows generated by a list-based heuristic. Nevertheless, energy consumption and reliability are not undertaken in this approach.

Furthermore, in [220], Alla et al. proposed a novel model for task scheduling based on Dynamic Dispatch Queues Algorithm (DDQA) and PSO rules. The proposed structure reduces the queue length and minimizes the waiting time of tasks. The makespan is minimized while resource utilization is maximized in this method. The proposed algorithm proves better than other current policies in cloud computing scenario in terms of load balancing. However, its scheduling time is long. Moreover, energy consumption, reliability, and monetary cost are not examined.

In [221], Haghighi et al. proposed a virtualization technique with regard to resource management. For this, a hybrid technique is proposed based on a micro-genetic algorithm and K-means for dynamic allocation of tasks. The proposed approach reduces makespan and offers a reasonable trade-off between the QoS of datacentres and energy consumption. Additionally, the number of VM migrations is reduced in this approach.

However, security, fault tolerance, reliability, and dynamic scheduling of workflows are not spotted.

Finally, Hemasian-Etefagh et al. [222] introduced a dynamic scheduling framework using an optimized version of WOA that represents a new conception of Grouping whales, called GWOA. GWOA is used as a cloud computing scheduler for the high workload to increase the throughput and response time while reducing the execution time in the cloud computing environment by pre-overcoming the premature convergence problem and pre-improving both the exploitation and exploration of the WOA. With a CEC2017 function set in MATLAB and the CloudSim simulator, the results indicate that GWOA achieves significant improvement rate in the execution time, response time, and throughput compared to other four existing algorithms. However, other parameters, such as energy consumption, SLA violation, computational cost, and migration, are not involved in the optimization of scheduling.

Artificial intelligence (AI)-based: AI-based scheduling is a highly-technical and specialized methodology that supports the creation of an intelligent technique working and reacting, like humans, to schedule and assign resources with different aspects, including intelligent and autonomous systems, nature-inspired intelligent systems, operational research systems, agent-based systems, neural networks, Machine Learning (ML), and expert systems [223]. In AI, the failure and error rates are virtually zero, as well as greater accuracy and precision are guaranteed for the resource allocation and scheduling in the cloud framework.

An ML algorithm runs on computer systems in order to learn complex relationships and make the right decisions accordingly. In cloud computing, ML algorithms have been evoked to predict the resources' status based on their future load and security. Kumar and Patel [224] used an ANN-based PSO model to map the incoming requests to resources using an ANN model; and to get the job done faster, PSO was used as a scheduling algorithm. The proposed mechanism achieves the goals of increasing reliability and availability, minimizing monetary cost, computational cost, and makespan, and handling the overloading problem. However, the ANN architecture needs improvements for making the network faster. Moreover, the PSO algorithm requires some modifications for minimizing the prediction error and increasing both the accuracy and the search speed.

With the increasing demand for cloud technology, the cloud load needs to be efficiently balanced for delivering a seamless QoS-based service to the different users of cloud. To tackle such issues, a novel hybridized load balancing technique based on the ANN and K-means methods was introduced by Negi et al. to calculate the load of an optimized VM in cloud systems based on the Back Propagation Network (BPN) methodology [225]. Further, these optimized VMs' loads are clustered into overloaded and underloaded by using an improved K-means method. Besides, a PSO-based task scheduling approach is used to assign dynamic tasks to the underloaded VMs. With the CloudSim tool, there is a significant improvement in cloud metrics, such as makespan, transmission time, and resource utilization. However, many other performance parameters are not regarded.

In [226], Gao et al. proposed an algorithm combined of: an improved GA, to find a set of optimal nodes used to execute the requested tasks in order to maximize the resource utilization; and a decentralized MOA, to minimize the bandwidth cost. When compared to NSGA-II and native GA, the proposed algorithm outpaces in terms of robustness, convergence time, and solution quality as the incoming tasks increases. However, only two algorithms are used for validating the proposed method, which increases doubts about the effectiveness of the proposed approach. Moreover, the flaws regarding network links (i.e., switches and routers), security, fault tolerance, scalability, and energy consumption are not well spotted.

ML has been widely employed for energy-aware task scheduling problems, particularly for predicting resource consumption as well as figuring out the schedule itself. In this regard, Sharma and Garg [227] proposed an ANN-based scheduler that, for a given task, predicts the best computing resources by taking the current cloud environment state and the incoming task as two inputs. GA is used to generate a large dataset of up to 18 million training instances used for training the neural network based on the BPN, resulting in a 99.9% output accuracy. The results have shown a decrease in makespan and energy consumption while reducing both the execution overhead and the number of active racks. However, workflow applications are not incorporated. Forecasting the temperature effects when making scheduling decisions are not taken into account, too.

Prediction-based: Prediction-based scheduling is associated with the behavior of methods and various measures while allocating resources. Sometimes, predicting the influential resource requirements and users' demand for the future, using automatic resource allocation or resource reservation techniques is deemed substantial for effective task scheduling and optimized resource allocation in the cloud environment [228,229].

Hu et al. [230] proposed a Prediction-based ACO classification algorithm (PACO) that acts based on the principle of task priority to schedule tasks to VMs, taking different QoS parameters into consideration. Accordingly, the proposed algorithm classifies the arriving jobs into two species: compute-intensive and network-intensive. The solution provided by PACO involves both the makespan and the computational cost with a more improved rate of about 3.33% than the ACO. However, PACO does not perform well if there are a large amount of hosts and VMs. Furthermore, the total cost and makespan needs more significant improvement. Furthermore, the impact of intensive tasks on the network is not taken into account.

Most Internet of Things (IoT) devices have sensors that generate a continuous stream of data that in turn calls for further research efforts into the cloud area. Within this framework, in [231], Vashishth et al. presented a predictive approach to task scheduling, aiming to increase the reliability and efficiency of the cloud system while processing big data on the cloud by using ML classifiers, such as K-Nearest Neighbor (KNN), Random Forest (RF), and Naïve Bayes, as tools for predicting the VM best suited for a task in a testing dataset. Herein, the PSO is used to generate a dataset used to train the classifiers. The proposed scheme performs the task allocation job almost 10 times faster than the traditional algorithms, in terms of processing time and allotment time; however, the PSO algorithm outpaces in terms of resource utilization. As a drawback, workflow scheduling is not included in this work.

For federated cluster environments, Gabaldon et al. [232] presented a novel approach combining PSO and Fuzzy-based GA (MPSO-FGA) to solve a scheduling problem with parallel applications, aiming to minimize both the makespan and total energy consumption. This algorithm is characterized by a weighted blacklist used to effectively represent the resource heterogeneity, computational resource availability, communication resources dispute, and the application requirements. However, scalability and throughput are not minded.

Finally, Li et al. [233] proposed a time-series-based fluctuation-aware workflow scheduling approach based on the prediction theory. In their study, the fluctuant (time-varying) VMs' performance is taken up, and an Auto-Regressive Moving Average model (ARIMA) is subsequently properly used to predict VMs' future performance, on which the GA depends to perform cost-effective schedules at runtime, thereby achieving lower makespan and low SLA violation rate. Nevertheless, energy consumption, scalability, and reliability are not analyzed in this approach.

5.3.2. Summary of the scheduling approaches reviewed based on the task-resource mapping scheme

Some of the selected articles have been studied and constructively commented in the previous subsection, along with their advantages and weaknesses, based on the task-resource mapping scheme adopted

in scheduling. A summary comparison between those articles, featuring their most notable merits and inevitable demerits, is illustrated in Table 8.

5.4. Scheduling constraint

Deadline, priority, budget, and fault tolerance constraints are significant factors in the cloud scheduling field as they might have a negative influence on the SLA if there are a large number of applications unable to meet these constraints. In the following, the selected mechanisms in this regard are discussed, along with their key traits.

5.4.1. Overview of the selected mechanisms

Deadline: In [200], Rodriguez and Buyya evoked the issue of resource scheduling and provisioning to fulfill the resource management demands required by scientific workflows in cloud systems. PSO metaheuristic optimization technique is used for minimizing the total execution time while meeting the user-defined deadline constraint. The proposed approach incorporates basic cloud principles, such as heterogeneity, scalability, and resource utilization. However, this work lacks parallel computations. Furthermore, availability, reliability, and energy consumption are not validated in the proposed model.

Visheratin et al. [247] put forward a new hybrid algorithm, LDD-LS (Levelwise Deadline Distributed Linewise Scheduling) and IaaS Cloud Partial Critical Paths (IC-PCP) algorithm, to initialize CDCGA (Cloud Deadline Coevolutional Genetic Algorithm) for scheduling scientific workflowa in hard-deadline constrained clouds under user-specified QoS constraints, like cost or makespan. The major goal of LDD-LS is to make workflow lines and calculate each line's computational time. Consecutively, the overall deadline is set across the workflow levels. Though, two different heterogeneous populations are generated in CDCGA. One population represents the sorted list of task identifiers, whereas the other population includes the sorted list of computational capabilities of resources, concluding that the CDCGA implementation is better than LDD-LS; however, it is possible to observe that throughput, scalability, and fault tolerance are not anaylzed in this scheme.

Wu et al. [248] proposed two deadline-constrained algorithms, Probabilistic Listing (ProLis) as well as an L-ACO, to minimize the makespan of workflow scheduling in the cloud environment. With the help of the ProLis algorithm, a deadline is distributed to each task, the tasks are ranked, and each task is sequentially allocated its adequate resources so as to be executed according to the QoS requirements. Moreover, ACO is employed by L-ACO to build various task-order lists for finding effective scheduling solutions that minimize the makespan under the deadline constraint and at the lowest cost. However, both the performance variation and start-up/boot time of VMs are not undertaken in this study.

Maurya and Tripathi [249] applied, besides two heuristic algorithms, two meta-heuristic algorithms, PSO and CSO, to the BoT and workflow scheduling problems for minimizing the makespan and execution cost of applications in cloud systems while meeting deadline constraints. As overall, the CSO algorithm performs better than other algorithms in terms of cost optimization. However, real cloud computing environments need to be triggered in the future for more reliability. Furthermore, neither energy consumption, scalability, fault tolerance, overhead, nor availability are investigated in this approach.

Priority: A Bi-Criteria priority-based PSO (BPSO) was proposed by Verma and Kaushal, taking the deadline and budget constraints into account to reduce the makespan and execution cost while scheduling workflow tasks in clouds [250]. The bottom level defined in the HEFT algorithm is used to assign priority for each workflow task so that the PSO can perform then according to these priorities. All particles have fitness values used to constantly indicate their performances and velocities in order to keep track of the particles' flight more realistically. The optimization process utilizes the experience of the neighboring particles, too. The proposed algorithms have shown a promising performance. However, only two algorithms, Budget Constrained HEFT (BHEFT) and the standard PSO, are used as counterparts for performance evaluation purposes. In addition, workload, reliability, and energy consumption are not minded.

Milan et al. [251] proposed a scheduling algorithm based on BFO to reduce the idle time of VMs. In the proposed method, each bacterium generated by the greedy algorithm shows a solution that indicates the best position as well as represents the bacteria cost. So, tumble and move are the two main steps to be used, and bacteria eventually follow these steps during the foraging process. The proposed algorithm reduces the running time, makespan, energy consumption, and degree of imbalance. However, this method is still not ideal enough for reducing the makespan and energy consumption. In addition, reliability is not regarded.

Budget: Verma and Kaushal [252] relied on a Budget Constrained Priority-based GA (BCHGA) to perform workflow scheduling over the cloud resources in order to, within a predefined budget, optimize the execution costs. For increasing the diversity of the population, priority is assigned to each task in the workflow using bottom level and top level approaches. The proposed algorithm is evaluated based on realistic workflows and it shows more promising performance than the standard GA, the only algorithm used in the comparison. This method does not touch the metrics of scalability, reliability, and energy consumption.

Wang et al. [253] proposed a budget-constrained scheduling algorithm based on PSO for scheduling of workflows in the cloud environment. In this algorithm, the makespan is minimized while preserving the user's budget. AS the number of iterations and particles increases, the algorithm will be more reliable to achieve better performance. However, finding the best solution might take a long time. Moreover, availability, scalability, energy consumption are not evaluated.

Fault tolerance: Guo and Xue [254] discussed a Cost-Effective Fault-Tolerant (CEFT) scheduling algorithm that should meet a given deadline in cloud systems. In the case of hardware failure, either permanent or transient fault tolerance is provided by this approach to tasks by applying the Primary/Backup (P/B) approach, which contains two duplicated copies of each task. Herein, the tasks are independent of each other and do not carry any precedence order, as well as the iterative method is followed in this approach for optimizing the proposed resource allocation technique more effectively. In every iteration, VM is selected for the upcoming task by using the PSO algorithm. The iterative approach is used in this method for reducing the execution cost of the task. Also, reliability is guaranteed. Nevertheless, this scheme might cause overheads to the system as well as require much computational resources to be available because of its iterative nature. Additionally, makespan, scalability, and energy consumption are not evaluated in this work.

Abdulhamid et al. [255] devised a CheckPointed LCA (CPLCA) mechanism for task allocation in the cloud system. The authors use a task migration scheme to handle the execution failures of independent tasks. CPLCA is used to schedule the failed tasks on an underloaded VM. Minimal makespan, minimal average response time, secure fault tolerance, and reliability metrics are produced by the CPLCA scheme. Furthermore, in the event of failure, restarting the task from the initial point is avoided by using a checkpointing technique that saves the state of a task at regular intervals. However, the task execution is suspended at the time of this regularly repeated checkpointing process, thereby increasing the overhead and reducing the throughput of the system.

Abdulhami et al. [256] presented a fault-tolerant dynamic clustering LCA (DCLCA) approach to the scheduling of independent tasks in the cloud. The objective of this technique is fault reduction in the case of job failure. The devised technique is designed on the basis of using elasticity and resource utilization as QoS objectives. This technique uses Kafka technology, replication protocol, and reactive mechanism. The experimental results validate the superiority of the proposed strategy in terms of reducing the fault rate and makespan. This technique has a

Table 8
Categorization of the techniques reviewed based on the task-resource mapping scheme, along with their advantages and weaknesses.

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Static						
Babu and Krishna [214]	ABC *Meta-heuristic (swarm)*	Weighted RR (WRR), FCFS, and Dynamic Load Balancing (DLB) [234,235] *Heuristic*	Independent	• Low makespan • High throughput • Low queuing time • High priority fulfillment	• Low scalability • Low reliability	Simulation (CloudSim)
Matos et al. [215]	Hybrid (GA with MR) *Meta-heuristic (evolutionary) and heuristic*	FCFS and PSO [155,199] *Heuristic and meta-heuristic (swarm)*	Workflow scheduling	• Low makespan	• Low scalability • High computational cost	Simulation (CloudSim)
Dynamic						
Islam and Habiba [218]	ACO and VNPSO *Meta-heuristic (swarm)*	SARDIG [236] and SAREG [237] *Heuristic*	Independent	• High security • High throughput • Low monetary cost • Minor deadline violation	• Low reliability	Simulation (CloudSim)
Rahman et al. [219]	Hybrid (GA and DCP) *Meta-heuristic (evolutionary) and heuristic*	Min-Min, Max-Min, HEFT, Myopic [238], GRASP [239], and GA *Heuristic and meta-heuristic (evolutionary)*	Workflow scheduling	• Low makespan • High fault tolerance	• Low reliability	Simulation (GridSim)
Alla et al. [220]	Hybrid (PSO with DDQA) *Meta-heuristic (swarm) and heuristic*	FCFS and PSO *Heuristic and meta-heuristic (swarm)*	Independent	• Low makespan • High resource utilization • High load balancing	• High scheduling time • Low reliability	Simulation (CloudSim)
Haghighi et al. [221]	Hybrid (GA with K-means) *Meta-heuristic (evolutionary) and ML*	PSO and GA *Meta-heuristic (swarm and evolutionary)*	Independent	• Low makespan • Low energy consumption • Minor SLA violation • Low VMs migrations	• Low reliability	Simulation (CloudSim)
Hemasian-Etefagh et al. [222]	GWOA *Meta-heuristic (swarm)*	WOA, improved WOA (CWOA), PSO, and Bat Algorithm (BA) *Meta-heuristic (swarm)*	Independent	• Low makespan • High throughput	• High energy consumption • Terrible SLA violation • High computational cost	Simulation (CloudSim)
Artificial intelligence based						
Kumar and Patel [224]	Hybrid (PSO with ANN) *Meta-heuristic (swarm) and ML*	N/A *N/A*	Independent	• Low makespan • Low monetary cost • Low computational cost • High reliability	• Low makespan • Low-performance ANN architecture • Low PSO-based search speed	Simulation (N/A)
Negi et al. [225]	Hybrid (PSO with ANN and K-means) *Meta-heuristic (swarm) and ML*	Max-Min, Min-Min, RR, and FCFS *Heuristic*	Independent	• Low makespan • High resource utilization	• Low reliability	Simulation (CloudSim)
Gao et al. [226]	Hybrid (GA with MAO) *Meta-heuristic (evolutionary)*	GA and NSGA-II *Meta-heuristic (evolutionary)*	Independent	• Low makespan • High resource utilization • Low bandwidth	• Low reliability • Low scalability	Simulation (N/A)
Sharma and Garg [227]	Hybrid (GA with ANN) *Meta-heuristic (evolutionary) and ML*	MinMIN-MINMin [240], linear regression [241], and GA *Heuristic and meta-heuristic (evolutionary)*	Independent	• Low makespan • Low energy consumption	• Low scalability • Low reliability	Simulation (CloudSim with MATLAB)
Prediction based						
Hu et al. [230]	PACO *Meta-heuristic (swarm)*	ACO *Meta-heuristic (swarm)*	Independent	• Low makespan	• Low scalability	Simulation (CloudSim)
Vashishth et al. [231]	PSO *Meta-heuristic (swarm)*	KNN, RF, and Naïve Bayes *ML techniques*	Independent	• High resource utilization	• High makespan	Simulation (CloudSim and Java-ML)
Gabaldon et al. [232]	Hybrid (MPSO with FGA) *Meta-heuristic (swarm and evolutionary)*	FCFS, CBS [242], JPR [243], Min-Min, MOGA [244] *Heuristic*	Workflow scheduling	• Low makespan • Low energy consumption	• Low scalability • Low throughput	Simulation (GridSim)
Li et al. [233]	Time-series-based GA *Meta-heuristic (evolutionary)*	PSO + ARIMA [245], Non-predictive PSO [200], and Non-predictive GA [165,246] *Meta-heuristic (swarm and evolutionary)*	Workflow scheduling	• Low makespan • Low monetary cost • Minor SLA violation	• Low fault tolerance • Low scalability • High complexity	Real environment

Table 9
Categorization of the techniques reviewed based on the scheduling constraint(s), along with their advantages and weaknesses.

Research	Technique(s) applied *Type of algorithm*	Compared against *Type of algorithm*	Nature of tasks	Advantages	Weaknesses/challenges	Testing environment
Deadline						
Rodriguez and Buyya [200]	PSO *Meta-heuristic (swarm)*	IC-PCP [197] and Scaling Consolidation Scheduling (SCS) [257] *Heuristic*	Workflow scheduling	• Minor deadline violation • High scalability • High resource utilization	• Low reliability • Low elasticity	Simulation (CloudSim)
Visheratin et al. [247]	CDCGA *Meta-heuristic (evolutionary)*	IC-PCP [197] and LDD-LS *Heuristic*	Workflow scheduling	• Minor deadline violation • Low makespan • Low monetary cost	• Low efficiency	Real environment
Wu et al. [248]	ACO and ProLis *Meta-heuristic (swarm) and heuristic*	IC-PCP [197] and PSO [200] *Heuristic and meta-heuristic (swarm)*	Workflow scheduling	• Minor deadline violation • Low makespan • Low monetary cost	• Low scalability	Simulation (Java)
Maurya and Tripathi [249]	PSO and CSO *Meta-heuristic (swarm)*	IC-PCP [197] and SCS [257] *Heuristic*	Independent and workflow scheduling	• Minor deadline violation • Low makespan • Low monetary cost	• Low reliability	Simulation (CloudSim)
Priority						
Verma and Kausha [250]	BPSO *Meta-heuristic (swarm)*	BHEFT [258] and PSO *Heuristic and meta-heuristic (swarm)*	Workflow scheduling	• High priority fulfillment • Minor deadline violation • Low makespan • Low monetary cost	• Low load balancing	Simulation (Java)
Milan et al. [251]	BFO *Meta-heuristic (swarm)*	PSO [259], hybrid (GA and ACO) [260], Symbiotic Organism Search (SOS) [261], and GA [262] *Meta-heuristic (swarm and evolutionary)*	Independent	• High priority fulfillment • Low makespan • Low energy consumption	• Low reliability	Simulation (CloudSim)
Budget						
Verma and Kaushal [252]	BCHGA *Meta-heuristic (evolutionary)*	GA *Meta-heuristic (evolutionary)*	Workflow scheduling	• High priority fulfillment • Low monetary cost • Minor budget violation	• Low reliability	Simulation (Java)
Wang et al. [253]	PSO *Meta-heuristic (swarm)*	Multi-objective List Scheduling (MOLS) [263] *Heuristic*	Workflow scheduling	• Minor budget violation • Low makespan • High scalability	• High response time	Simulation (Java)
Fault tolerance						
Guo and Xue [254]	PSO (CEFT) *Meta-heuristic (swarm)*	Min-Min, CEFTNI (CEFT without Iteration), and CEFTNR (CEFT without Rescheduling) *Heuristic and meta-heuristic (swarm)*	Independent	• High fault tolerance • Low monetary cost • Minor deadline violation • Low makespan	• Low resource utilization • High overhead	Real environment
Abdulhamid et al. [255]	CPLCA *Meta-heuristic (evolutionary)*	ACO, LCA and GA *Meta-heuristic (swarm and evolutionary)*	Independent	• High fault tolerance • Low makespan • Low response time • High security • High reliability	• High overhead • Low throughput	Simulation (CloudSim and CloudAnalyst)
Abdulhamid et al. [256]	DCLCA *Meta-heuristic (evolutionary)*	Max-Min [264], MTCT [265], ACO [266], and NSGA-II [267] *Heuristic ad meta-heuristic (swarm and evolutionary)*	Independent	• High fault tolerance • Low makespan • High elasticity • High resource utilization	• Low reliability	Simulation (CloudSim)

major issue; that is, it does not consider the throughput of the system, execution cost, energy consumption, and reliability.

5.4.2. Summary of the scheduling approaches reviewed based on the scheduling constraint

Some of the selected articles have been studied and constructively commented in the previous subsection, along with their advantages and weaknesses, based on the scheduling constraint(s) adopted in the study. A summary comparison between those articles, featuring their most important merits and inescapable demerits, is illustrated in Table 9.

6. Simulation tools and their comparison

Some experiments might compromise the end-user QoS; therefore, it is practically not easy to experiment new techniques in a real environment. Cloud computing has some prominent simulation tools meant for analyzing new scheduling algorithms as well as evaluating their performance in different scenarios over cloud environments. Fig. 12 depicts that CloudSim simulation toolkit is the popular tool most used for task scheduling, whose existing programmatic classes which can be extended in accordance with the algorithm requirements in order to eval-

Table 10
Comparison between simulation tools most widely used for task scheduling in clouds.

Tool	Platform	Language/ Script	Graphical support	Communication model support	Support windows environment	Energy consumption modelling support	Output result format	Availability
GridSim [268]	N/A	Java	No	Limited	Yes	No	Text	Open source
CloudSim [269]	SimJava (GridSim)	Java	Limited (via CloudAnalyst)	Limited	Yes	Limited	Text	Open source
CloudAnalyst [270]	Java SE, Swing, and SimJava (CloudSim)	Java	Full	Limited	Yes	Limited	PDF	Open source
EMUSim [271]	Emulation (AEF) and CloudSim	Java	Limited (via NEPI)	Limited	No	Limited	Text	Open source
NetworkCloudSim [272]	CloudSim	Java	No	Full	Yes	Yes	Text	Open source
WorkflowSim [273]	CloudSim	Java	No	Limited	Yes	Limited	Text	Open source
GreenCloud [107]	NS2	C+/OTcl	Limited (via Nam)	Full	No	Yes	Dashboard plots	Open source
GroudSim [274]	N/A	Java	Limited	No	Yes	No	Java API, Tracer handlers, and Filters	Open source
iCanCloud [275]	OMNET and MPI	C+	Full	Full	Yes	In progress	Text	Open source

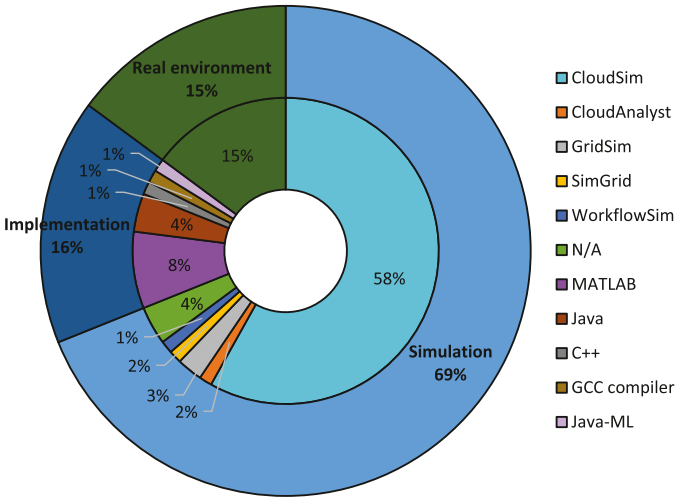


Fig. 12. Percentage of simulation, implementation, and real environment among the testing tools.

uate such diverse QoS parameters as makespan, monetary cost, computational cost, reliability, availability, scalability, energy consumption, security, and throughput, with includible constraints, like deadline, priority, budget, and fault tolerance, by following one of the different task-resource mapping schemes. Using a simulator for performance validation of a new algorithm has main advantages including:

- *Time efficiency:* Implementing applications requires much less time and effort.
- *Applicability and flexibility:* Minor programming and deployment efforts are required to model application services and test their performance in heterogeneous environments.

Prominent simulation tools available in cloud computing, such as GridSim, CloudSim, CloudAnalyst, EMUSim, NetworkCloudSim, WorkflowSim, GreenCloud, GroudSim, and iCanCloud, are summarized and compared in Table 10 in terms of different parameters including base platform, programming language, graphical support, communication model support, output result format, availability, and the support level provided by the simulator. Looking closely at Table 10, it can be observed that the graphical support in most of the simulators is limited, which would require more effort from researchers and practitioners in order to implement a new technique or approach. It is also noted that energy consumption modelling support is generally very limited, while absent in some simulators, which calls for more implementations in this respect to strongly bring about this feature in the current simulators, especially with the increasing demand of green computing. Moreover, it would be very beneficial to extend the format of the output results to include common formats (e.g., csv, xml, etc.). Finally, the most notable is that all simulators are open source, which enables scholars or practitioners to easily implement new features or improve existing ones in the simulation tools, in a response to the diverse requirements of the cloud community. As shown in Fig. 12, simulation toolkits (particularly, CloudSim) are used as a testing environment in almost 69% of the selected studies, whereas 16% of the studies depend on implementation. It can be also seen that 15% of testing are conducted in a real environment.

7. Analysis and discussions

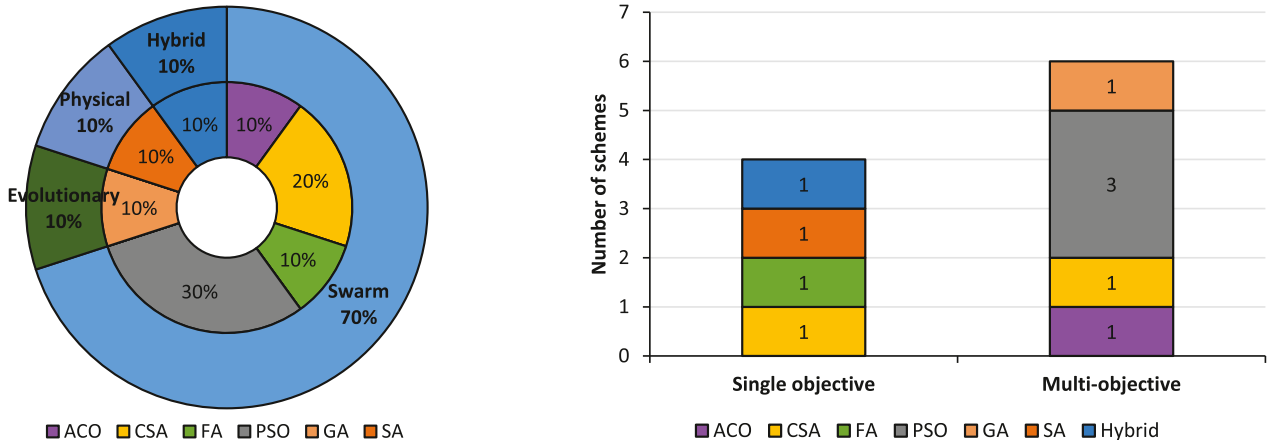
This section comprehensively summarizes and analyzes the meta-heuristic approaches for task scheduling in cloud computing. The discussions are held based on diverse aspects, including the presented taxonomy, types of algorithms, overall objectives and constraints, key sub-areas, practical impact, and strengths and limitations of the review.

7.1. The presented taxonomy

Meta-heuristic task scheduling approaches presented in Section 4 are analyzed in this subsection.

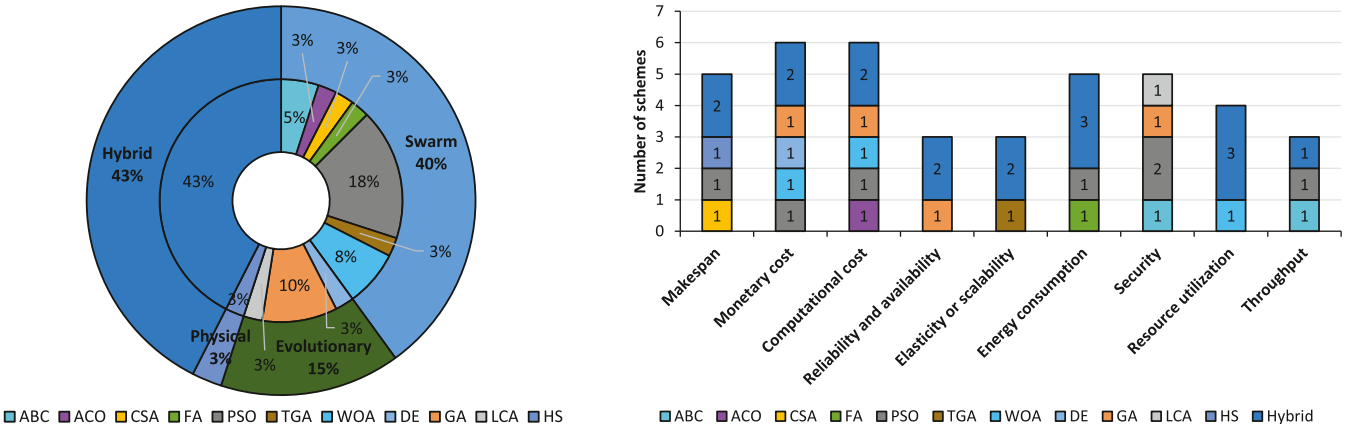
7.1.1. Nature of scheduling problem

Some of the selected studies have been reviewed in Section 5.1, based on the nature of the scheduling problem. Fig. 13 depicts the single and multi-objective approaches covered by these studies. As it is possible to observe, the most commonly used meta-heuristic methods of scheduling based on the nature of the scheduling problem are swarm-based (70%) in which PSO (30%) and CSA (20%) have the highest proportions as shown in Fig. 13(a). Evolutionary (10%), physical (10%), and hybrid (10%) algorithms have further equal weightage for the optimizing-scheduling process. ACO (10%), FA (10%), GA (10%), and SA (10%) have also a significant effect on the performance of the scheduling approaches from the perspective of single and multi-objective approaches. As depicted in Fig. 13(b), PSO was applied exclusively, most remarkably in multi-objective approaches. It is also noted that there are generally more scheduling schemes associated with multi-objective approaches



(a) Usage percentage in terms of the meta-heuristic algorithms. (b) Usage frequency in terms of the nature of scheduling problem.

Fig. 13. Usage ratio of the single and multi-objective approaches among the selected studies.



(a) Usage percentage in terms of the meta-heuristic algorithms. (b) Usage frequency in terms of the primary scheduling objective.

Fig. 14. Usage ratio of the QoS parameters (primary scheduling objectives) among the selected studies.

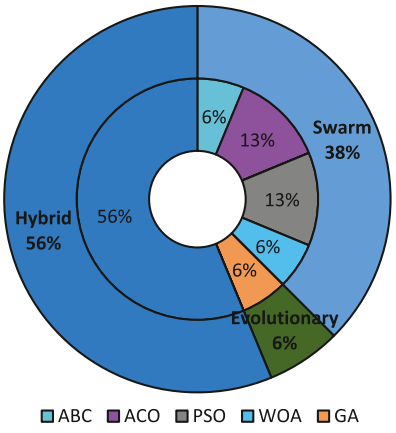
than single objective ones, thereby promoting further future incorporation of multi-objective techniques for more reliable scheduling of tasks in the cloud environment.

7.1.2. Primary objective of scheduling

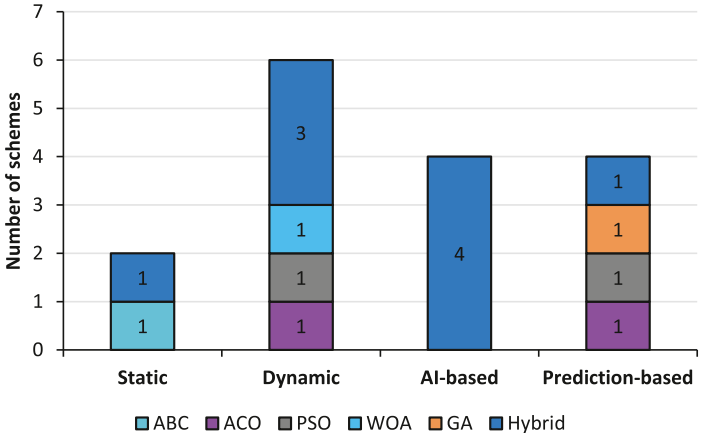
Some of the selected studies have been reviewed in Section 5.2, based on diverse primary scheduling objectives. Fig. 14 depicts the different scheduling objectives primarily considered in these studies. As it is possible to see, the most commonly applied meta-heuristic methods of scheduling based on a primary scheduling objective are hybrid algorithms (43%), as shown in Fig. 14(a), which are used most frequently for particularly achieving the two objectives of resource utilization and energy consumption as shown in Fig. 14(b). In the case of swarm-based techniques, note that the PSO (18%) was applied most widely with the security objective. Further weightage for the optimizing-scheduling process is given to swarm (40%), evolutionary (15%), and physical (3%) algorithms. As depicted in Fig. 14(b), makespan, monetary cost, computational cost, energy consumption, and security have more related schemes than other scheduling objectives. In order to generate higher profits, service providers take resource utilization seriously as a factor. Proper resource provisioning to the incoming applications could highly increase resource utilization. As discussed in the related taxonomy, the issues surrounding appropriate resource allocation and provisioning have been elaborated by many researchers. However, it is evident that resource utilization is closely related to the problem of increased energy consumption. To overcome this, merging of the workloads on the VMs might significantly reduce the energy consumption by either switching off the free VMs or dedicating them for use with new application demands. Thus, calculating future resource consumption may aid the reduction of VM migrations as well as bring further revenue to cloud service providers.

7.1.3. Task-resource mapping scheme

Some of the selected studies have been reviewed in Section 5.3 based on the task-resource mapping scheme. Fig. 15 depicts task-resource mapping schemes covered by these studies. Once again, hybrid algorithms (56%) are observed to seize the highest percentage among meta-heuristic scheduling approaches by paying the most attention to the task-resource mapping schemes, as shown in Fig. 15(a). Further weightage for the optimizing-scheduling process is given to swarm (38%) and evolutionary (6%) algorithms. ACO (13%) and PSO (13%) in swarm-based techniques as well as GA (6%) in evolutionary techniques have a significant relative effect on the performance of the scheduling approaches from the perspective of task-resource mapping schemes. It is good to mention that researchers have not triggered physical algorithms in this regard. As depicted in Fig. 15(b), dynamic scheduling has the greatest usage, which indicates the effectiveness and efficiency of this scheme. It is also observed in this respect that, in general, hybrid algorithms are the most commonly used among meta-heuristic methods, but exclusively with AI-based scheduling schemes.

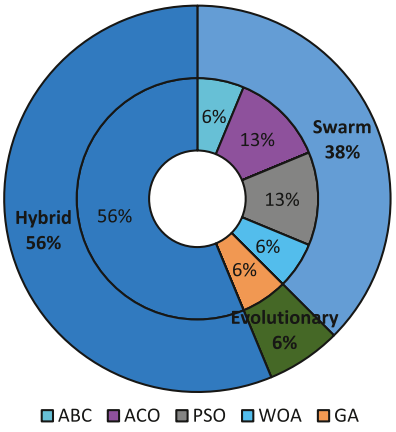


(a) Usage percentage in terms of the meta-heuristic algorithms.

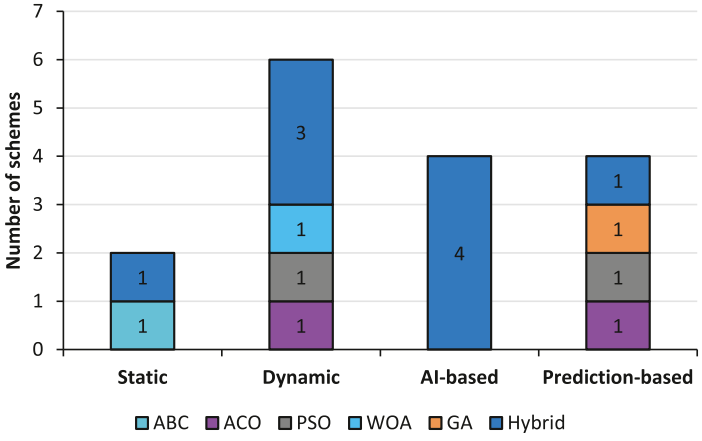


(b) Usage frequency in terms of the task-resource mapping scheme.

Fig. 15. Usage ratio of the task-resource mapping schemes among the selected studies.



(a) Usage percentage in terms of the meta-heuristic algorithms.



(b) Usage frequency in terms of the scheduling constraints.

Fig. 16. Usage ratio of the constrained approaches among the selected studies.

7.1.4. Scheduling constraint

Some of the selected studies have been reviewed in Section 5.4, based on the scheduling constraint in these studies. Fig. 16 depicts various constrained approaches. Once again, it is possible to notice that the constrained approaches mainly depend on swarm algorithms (67%) in which PSO (42%) has the highest proportion as shown in Fig. 16(a). Further weightage for the optimizing-scheduling process is given to the evolutionary algorithms (33%). GA (17%) and LCA (17%) in evolutionary algorithms as well as ACO, BFO, and CSO in swarm algorithms have the same effect (8%) on the performance of scheduling under constraints. It is worth mentioning that researchers, once again, have not evoked physical algorithms. As depicted in Fig. 16(b), deadline is the constraint most emphasized by meta-heuristic techniques, especially PSO. Moreover, it is clear that fault tolerance is also a widely adopted constraint.

7.1.5. Overall taxonomy

As shown in Fig. 17, the total number of approaches undertaking a primary scheduling objective (QoS parameter) represents 51% of the maximum meta-heuristic approaches addressed in this study. In the primary-objective-based scheduling techniques, the primary focus is on the monetary cost and computational cost with the same proportion of 8%, as well as makespan, energy consumption, and security with 6% for each. However, other objectives were paid less attention. Moreover, resource elasticity or scalability, system reliability, and throughput features are not considered by most of the work. It is worth mentioning here that the undesirable sequence of resource failures could be reduced by developing an efficient scheduling technique. The failure probability of a task or a resource can be minimal when making a scheduling decision, by integrating the scheduling framework with a failure prediction module based on such intelligent methods as data mining, statistics-based, ML-based, etc.

Furthermore, Fig. 17 reveals that deadline (6%) and fault tolerance (4%) are the most frequently adopted constraints in the problem of cloud scheduling. For deadline constrained applications, users would prefer reducing the execution time. Similarly, the fault tolerance needs to be maximal in mission-critical applications or systems.

7.2. Types of algorithms

Meta-heuristic techniques are still in their early stage as the commonly used techniques in the literature, such as ACO, PSO, GA, etc., are yet to be duly utilized to determine the best possible fit in the cloud task scheduling paradigm. Our selected studies aim mainly at empowering the knowledge of researchers by providing them with some basic concepts and advances in the field. It is not an easy task to solve a particular scheduling problem using a meta-heuristic technique, instead, the user's choice and expertise can be better exploited. Categorization of diverse task scheduling meta-heuristic techniques in clouds into swarm, evolutionary, physical, emerging, and hybrid approaches, along with their

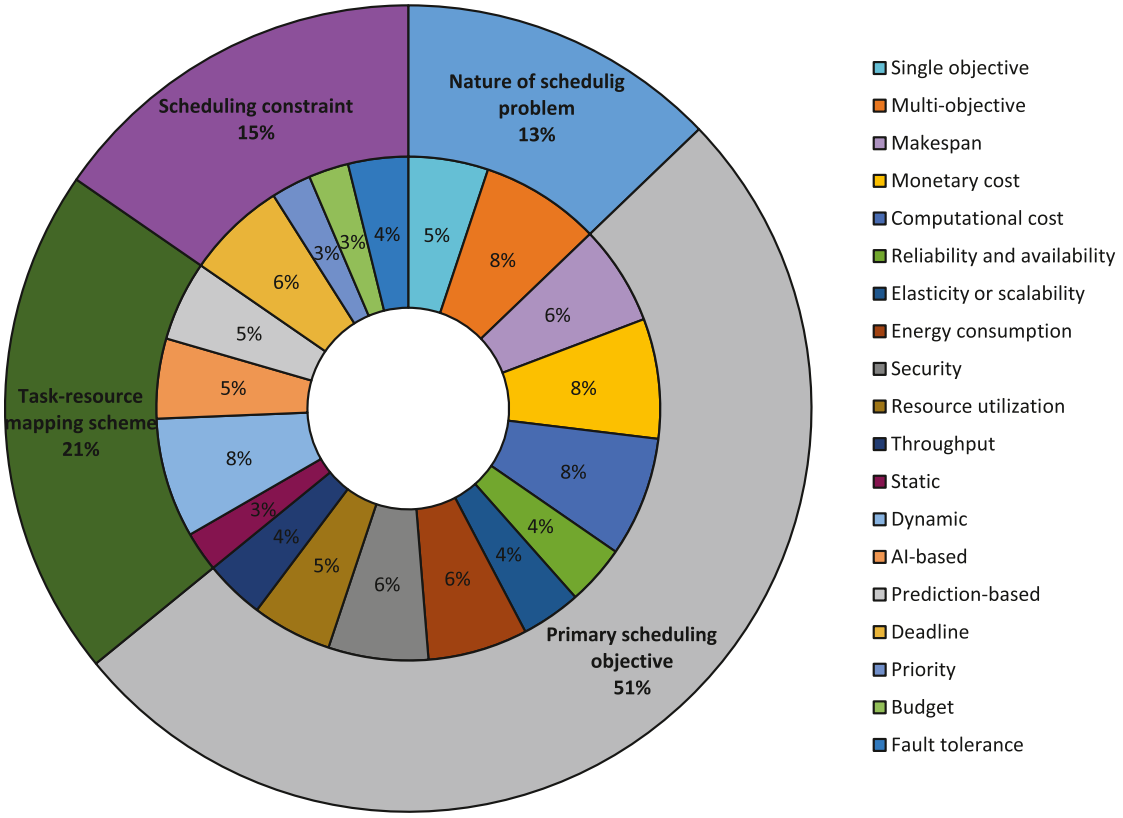


Fig. 17. Overall usage ratio of the diverse approaches to cloud task scheduling among the selected studies.

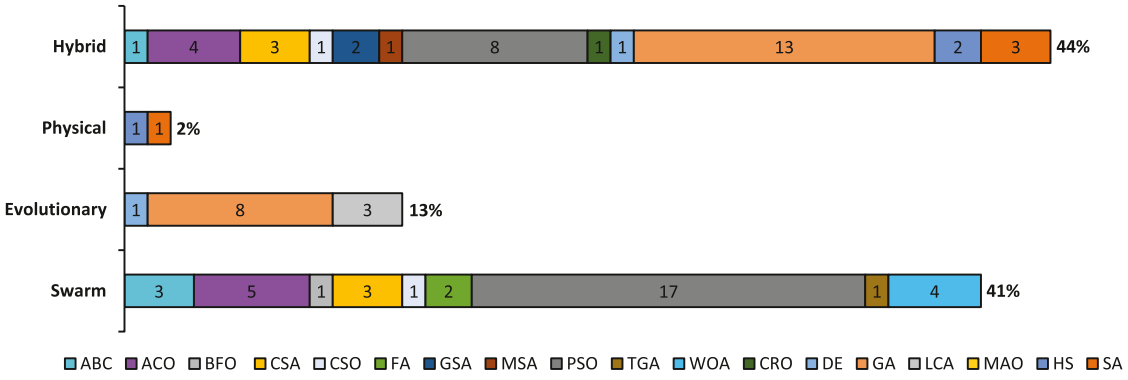


Fig. 18. Overall usage ratio of the meta-heuristic algorithms covered by the selected studies.

relevant overall usage proportions are depicted in Fig. 18. Speaking of the hybrid and swarm techniques whose major close total usage ratios are 44% and 41%, respectively, PSO is remarked as the most active algorithm among them. Sometimes, other meta-heuristic or even heuristic approaches are hybridized with those algorithms. The search strategy and convergence speed of meta-heuristic techniques differs, depending on the fact that the system overall performance can be further boosted using a hybrid version. Similarly, in evolutionary techniques (13% overall usage), most researchers applied the GA. Considering physical techniques (13% overall usage), no approach is dominant; however, the HS and SA have low, equal weightages. As it is also possible to observe in Fig. 18, PSO and GA have been applied the most among meta-heuristic scheduling approaches in the cloud, independently and in hybridization. It is also very notable that the emerging meta-heuristic algorithms had much less attention by researchers in all taxonomy aspects addressed in this study.

7.3. Overall objectives and constraints

The primary objectives and popular constraints covered by the task scheduling approaches presented in the selected papers have been triggered within Section 5. Table 11 illustrates a side-by-side assessment of those approaches based on the *overall* involved objectives (QoS parameters) and constraints to be considered while scheduling. Findings in Table 11 reveal that makespan as an objective and deadline as a constraint are comparable keys in the meta-heuristic approaches for task scheduling in clouds. However, the measures of reliability, scalability, security, computational cost, and budget were not deemed necessary in some studies despite their significant impact on the cloud service. All in all, according to Table 11, in which all selected studies are illustrated, the highest and lowest scores of each involved technique are also apparent. It is interesting to notice that the highest score is shared between makespan and monetary cost as objectives as well as deadline as

Table 11
A side-by-side comparison between the presented scheduling approaches based on the objectives and constraints of scheduling.

Algorithm(s)	Reference	Objectives (QoS parameters)									Constraints			
		Makespan	Monetary cost	Computational cost	Reliability and availability	Elasticity or scalability	Energy consumption	Security	Resource utilization	Throughput	Deadline	Priority	Budget	Fault tolerance
Swarm														
ABC	[190]	√	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗
	[195]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
	[214]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	√	✗	✗
ACO	[152]	√	√	✗	✗	✗	✗	✗	√	✗	√	✗	✗	✗
	[170]	√	✗	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[218]	✗	√	✗	✗	✗	✗	√	✗	√	√	✗	✗	✗
	[230]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[248]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
BFO	[251]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	√	✗	✗
CSA	[148]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[154]	√	√	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
CSO	[162]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
	[249]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
FA	[148]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[185]	✗	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
PSO	[155]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
	[151]	√	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
	[153]	√	√	✗	✗	✗	√	✗	√	✗	✗	✗	✗	✗
	[161]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
	[166]	✗	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[170]	√	✗	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[182]	√	√	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
	[187]	✗	√	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗
	[189]	√	√	√	✗	✗	✗	√	√	✗	✗	✗	✗	✗
	[150]	√	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
	[200]	✗	✗	✗	✗	√	✗	✗	√	✗	√	✗	✗	✗
	[218]	✗	√	✗	✗	✗	✗	√	✗	√	√	✗	✗	✗
	[231]	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
	[249]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
	[250]	√	√	✗	✗	✗	✗	✗	✗	✗	√	√	✗	✗
	[253]	√	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	√	✗
	[254]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	√
TGA	[180]	✗	✗	√	✗	√	✗	✗	✗	✗	✗	✗	✗	✗
WOA	[222]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
	[194]	✗	√	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗

(*continued on next page*)

Table 11 (*continued*)

Algorithm(s)	Reference	Objectives (QoS parameters)									Constraints			
		Makespan	Monetary cost	Computational cost	Reliability and availability	Elasticity or scalability	Energy consumption	Security	Resource utilization	Throughput	Deadline	Priority	Budget	Fault tolerance
Evolutionary														
DE	[167]	√	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
GA	[151]	√	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
	[165]	✗	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
	[170]	√	✗	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[173]	✗	✗	√	√	✗	✗	✗	√	✗	✗	✗	✗	√
	[188]	√	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗
	[233]	√	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
	[247]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
	[252]	✗	√	✗	✗	✗	✗	✗	✗	✗	✗	√	√	✗
LCA	[186]	√	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗
	[255]	√	✗	✗	√	✗	✗	√	✗	✗	✗	✗	✗	√
	[256]	√	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗	√
Physical														
HS	[163]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
SA	[148]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Hybrid														
ABC and PSO	[184]	√	✗	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
ACO with CRO	[168]	√	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗
ACO and CSA	[160]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
ACO and GA	[176]	√	✗	✗	√	✗	✗	✗	√	✗	✗	✗	✗	√
ACO with GSA	[193]	√	✗	✗	✗	✗	✗	✗	√	√	✗	✗	✗	✗
CSA with GD	[196]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
CSA with HS	[179]	√	√	√	✗	√	√	✗	✗	✗	✗	✗	✗	✗
ECS with GA	[181]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
GA with ANN	[227]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
GA with CLPS	[183]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
GA and DCP	[219]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	√
GA with fuzzy	[191]	√	✗	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
GA with HS	[164]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
GA with K-means	[221]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
GA with MAO	[226]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
GA with MR	[215]	√	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
GSA and GA	[172]	✗	✗	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
MSA and DE	[149]	√	✗	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗
PSO with ANN	[224]	√	√	√	√	✗	✗	✗	✗	✗	✗	✗	✗	✗
PSO with ANN and K-means	[225]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
PSO with DDQA	[220]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
PSO with GA	[232]	√	✗	✗	✗	✗	√	✗	✗	✗	✗	✗	✗	✗
PSO with GA and SA	[169]	✗	√	✗	✗	✗	√	✗	✗	√	✗	✗	✗	✗
PSO and Max-Min	[192]	√	✗	✗	✗	✗	✗	✗	√	✗	✗	✗	✗	✗
PSO with NN	[171]	√	√	√	✗	✗	√	✗	√	✗	✗	✗	✗	✗
SA with CSO	[177,178]	√	√	✗	✗	√	✗	✗	✗	√	✗	✗	✗	✗
SA with TS	[174,175]	✗	✗	✗	√	√	✗	✗	✗	✗	✗	✗	✗	✗

a constraint, whereas the lowest score is shared between reliability as an objective and budget as a constraint, for all selected studies. Finally, it can be concluded that, for effective cloud task scheduling, all factors are vital and crucial.

From the previous two subsections, it implies that more research is required and optimal solutions to diverse scheduling problems can be further found out based on the referred techniques. This precise investigation provides researchers interested in the area of cloud scheduling with a roadmap, along with a guideline on how to effectively improve the cloud service using meta-heuristic techniques.

7.4. Key sub-areas

To accustom the art followed by authors in order to show up their works, we have thoroughly, while writing this paper, analyzed many papers from different fields and journals, which eventually emerged this literature work. Literature prominence was brought to this work through various research perspectives by categorizing previous studies into five distinct sub-areas with respect to:

i) meta-heuristic scheduling algorithms,
ii) nature of scheduling problem,
iii) primary objective of scheduling,
iv) task-resource mapping scheme, and
v) scheduling constraint.

Further, various meta-heuristic algorithms for task scheduling are categorized in Section 4.3 into swarm, evolutionary, physical, emerging, and hybrid algorithms as well as comprehensively summarized in Table 11. However, the focus of attention in this survey is categorizing diverse task scheduling approaches from different perspectives, including the nature of the scheduling problem, the primary objective of scheduling, the task-resource mapping scheme, and the scheduling constraint, which have been discussed in Section 5, as well as comprehensively summarized in Tables 6–9, respectively. Also, open challenges related to the respective spheres of research are highlighted. However, issues regarding task failure prediction, temperature, bandwidth/speed, load balancing, and storage while task scheduling are not triggered in this work due to the limited length of the paper.

7.5. Analysis of practical impact

During this survey, one of the challenging tasks was the analysis of the effectiveness of existing researches. However, their practical impact has been taken very seriously by looking at the quality rank (quartile) assigned by Scopus to each journal. Table 12 shows the distribution of the selected articles in the literature per top 24 ranked journals. Additionally, publisher, quartile rank, as well as the country of each journal are listed. Compared to other journals, the ASCJ involves most authors in the area of task scheduling. It is also notable that most of the studies in the task scheduling domain were published in diverse major conferences. Among the 71 recent technical studies picked from the literature for this study, 32 articles were published in major journals as shown in Table 12.

7.6. Strengths of the review

This research process strives to supply further useful information about a reasonable number of diverse task scheduling approaches under the umbrella of cloud computing, along with their merits and demerits, as illustrated in tables of Section 5 (Tables 6–9 and Table 11). In addition, the discussions associated with them, being the limelight of this review, would empower the researchers' knowledge of various task scheduling strategies in the cloud area. Moreover, the period from 2011 to 2020 was surveyed in order to really show up the volume of work done in the different approaches comprehensively.

7.7. Limitations of the review

Conducting a high-quality systematic review is the main objective of this study; however, some limitations were imposed. Hence, after collecting and analyzing the relevant data on task scheduling in cloud computing, the limitations of this review were spotted as follows:

- *Research validity:* To rely heavily on Google Scholar, among many others, as an electronic database does not significantly guarantee that all selected studies are applicable to data extraction, for a variety of reasons, for example, searching with wrong or irrelevant keywords.
- *Approach scope:* Only the meta-heuristic algorithms are mainly focused as prominent approaches to tackle the NP-hard problem of task scheduling in clouds.
- *Differentiation:* While searching different scholarly databases, which meta-heuristic approach as well as what QoS parameters perform the best are not clearly defined.
- *Comprehensiveness:* Not all constraints and QoS parameters are addressed (e.g., load balancing, storage, bandwidth, etc., are not considered). Moreover, other promising aspects, such as realistic applications, are not duly analyzed.
- *Evaluation of techniques:* No algorithm exact accuracy is clearly defined.

8. Open issues and challenges

On the basis of pay-per-use, cloud computing has achieved a lot of progress with regard to the implementation of scalable computing infrastructures. However, many current issues and challenges still need both more analysis and discussion. Looking closely at existing research into cloud task scheduling, generally, this area has diverse open issues still pending. Based on the existing literature, open challenges and issues surrounding task scheduling in cloud computing can be identified as follows:

8.1. Resource scheduling

Cloud resource scheduling includes such diverse challenges as heterogeneity, uncertainty, and dispersion of resources, which cannot be resolved by using traditional resource management approaches. Therefore, a large focus and priority should be paid to these cloud features in order to make cloud applications and services more reliable. Resource scheduling aims to map the right workloads, at the right time, to the most suitable resources so the resources are utilized in the best way by those applications in a balanced manner. In other words, workload should take up the minimal amount of resources that will, however, be well utilized in order to get the minimal task length (maximize the throughput of the system) while maintaining a desirable QoS level. This area still needs more effort to be devoted by researchers into developing new convenient solutions.

8.2. Quality of service (QoS)

A service provider provisions the amount of resources required by a cloud service to fulfill its QoS requirements. Consequently, an SLA paradigm is designed to regularly detect SLA violations that in turn decide whether compensation or penalty is awarded. Thus, service providers are required to dynamically provision an adequate amount of resources to avoid or even mitigate SLA violations.

8.3. Service level agreements (SLAs)

In cloud computing, the interaction between the computing environment and the cloud consumer needs to be minimal while accomplishing, in accordance with the SLA, the QoS requirements described by users.

Table 12
Distribution of articles per journals.

No.	Publisher	Country	2018 Q factor	Journal name	Number of articles
1	Elsevier	Netherlands	Q1	Applied Soft Computing (ASC)	4
2	IEEE	United States	Q1	IEEE Acess (IA)	3
3	Elsevier	Netherlands	Q1	Future Generation Computer Systems (FGCS)	2
4	Elsevier	Netherlands	Q1	Engineering Science and Technology, an International Journal	1
5	IEEE	United States	Q1	IEEE Transactions on Cloud Computing	1
6	IEEE	United States	Q1	IEEE Transactions on Parallel and Distributed Systems	1
7	IEEE	United States	Q1	IEEE Systems Journal	1
8	Springer	Netherlands	Q1	Journal of Grid Computing	1
9	Elsevier	Netherlands	Q1	Knowledge-Based Systems	1
10	Public Library of Science	United States	Q1	PLoS ONE	1
11	Springer	Netherlands	Q2	Cluster Computing (CC)	3
12	Springer	Netherlands	Q2	Journal of Supercomputing (JS)	3
13	Springer	Germany	Q2	Arabian Journal for Science and Engineering (AJSE)	2
14	IEEE	China	Q2	China Communications	1
15	Wiley Online Library	United States	Q2	Concurrency Computation Practice and Experience	1
16	IET	United Kingdom	Q2	IET Communications	1
17	Inderscience	United Kingdom	Q2	International Journal of Grid and Utility Computing	1
18	Elsevier	United States	Q2	Journal of Parallel and Distributed Computing	1
19	Springer	Germany	Q2	Neural Computing and Applications	1
20	Elsevier	Germany	Q2	Soft Computing	1
21	Elsevier	United States	Q2	Sustainable Computing: Informatics and Systems	1
22	Springer	Netherlands	Q2	World Wide Web	1
23	Springer	Netherlands	Q3	Wireless Personal Communications (WPC)	2
24	Springer	Netherlands	Q3	International Journal of Parallel Programming	1

Thus, an autonomic infrastructure-based effective strategy which, in advance, detects SLA violation, is a research issue by which performance degradation could be avoided.

8.4. Self-management service

In this case, a cloud provider aims to allocate and release cloud resources so the deployment charge is minimal while fulfilling the Service Level Objectives (SLOs). These methods usually include: i) request management at each level by forecasting the required number of application instances using an application performance model (ML approach) specially created to fulfill the QoS requirements, ii) a performance model used to occasionally define resource requirements for the expected forthcoming demand, and iii) estimating of future resource requirements so that resources are assigned automatically later when requested by end-users. The proactive model is one of such self-managing methods that occasionally allocate resources by analyzing the expected demand before resources are even requested [276]. On the other hand, the reactive method responds also to various instant demands before accessing the demand periodic forecast [276].

8.5. Energy management

One of the major problems in cloud computing is improving the energy efficiency. It has been assessed that 53% of the entire operational spendings go to powering and refrigeration of datacentres. In 2006, in the USA, nearly 1.5% of the total energy produced were consumed by datacentres, as well as the yearly growth proportion is estimated at 18%. Therefore, decreasing energy consumption is an urgent task given to IaaS providers. Reducing energy cost in datacentres is not the only goal, but also government rules and environmental standards should be adhered. There are two other methods used to decrease power consumption by switching off the idle systems: energy-aware server consolidation and energy-efficient task scheduling. Existing scheduling techniques have a main issue to achieve an acceptable trade-off between application performance and energy consumption.

8.6. Dynamic scalability

For cloud workloads, resource scaling and scheduling are needed for boosting the performance of an application within deadline or budget constraints. How much cloud resources should be acquired, and how should they be allocated to and released from the computing activities for optimizing the application performance within the deadline or budget constraints? The capability of dynamic acquiring or releasing of resources in response to demand is referred to as dynamic scalability. In a datacentre, dynamic resource management should mainly aim to fight against the waste of resources arising from underutilization. In addition, the aims of such a process should include tackling the problem of high response time, which may result in overutilization as well as violation of SLA between the service providers and recipients. Initially, the cloud workload (e.g., batch data processing, application server, web server, file server, transactional database, etc.) should be thoroughly understood for designing such a successful cloud platform. Then, the cloud consumer applications should be designed carefully to boost the scaling advantage. Thus, processes like dynamic infrastructure scaling, minimizing the response time of flexible demand, and maximizing the throughput of requests, can all be together achieved. An IT resource is difficult to be well-prepared to fulfill its all processing requests. Overutilization and underutilization are two troublesome dilemmas surrounding IT resources as a result of the ongoing demand around cloud resources.

8.7. Reliability

Service provider faces a main challenge in determining the method of delivering a responsive service that fulfills the users' diverse requirements while meeting the QoS parameters. Amongst many others, reliability is one of the most challenging cloud computing issues as cloud service providers currently want to provide their services to end-users with high performance, high quality, and low processing time while making maximum profit.

8.8. Security

The resources in distributed environments are virtualized and diversificated; therefore, security became a complicated process in cloud computing. Accordingly, resource monitoring, resource management, and resource security have emerged as eminent issues in cloud computing. Discrete PSO method was used to tackle the impact of security threats on the scheduling of workflows [277]. In this approach, a cloud model is used to quantify the security of tasks and VM resources. In addition, the user's satisfaction degree with the security of the allocation process of tasks to the virtual resources is generally estimated by measuring the similarity of the security cloud. Thus, the objectives of completion time and cost are combined with security. Likewise, the scheduling strategy can be figured out in some way to protect and secure the confidential and private information of the submitted applications.

8.9. Scheduling based on emerging meta-heuristic approaches

Several optimization problems were worked out using different meta-heuristic techniques, such as Whale Optimization Algorithm (WOA) [140], Imperialist Competitive Algorithm (ICA) [278], Bat Algorithm (BA) [279], Harris Hawks Optimization (HHO) [280], Grey Wolf Optimizer (GWO) [281], and Wind Driven Optimization (WDO) algorithm [282]. For task scheduling in cloud computing, WOA [222], robust algorithms like ICA [283], as well as BA [284] have already been applied. Building on past experiences, HHO, GWO, and WDO algorithms may also have the potential to solve various cloud task scheduling problems. Consequently, these algorithms as well as other emerging prominent algorithms can be adapted more creatively to individually or collectively optimize such diverse scheduling objectives as energy optimization, load balancing, scalable VM migration, etc.

8.10. Others

Speaking of scheduling in clouds, users usually pay close attention to both the execution time and the service price for workflow applications. Service consumers aim to use resources at the minimum monetary cost and makespan. In [285] in which a workflow is scheduled in grid computing, a trade-off factor was developed to compromise the cost and execution time as per user's preference. In cloud environment, the importance of the trade-off factor is exponentially increasing, especially with the presence of the heterogeneity and scalability of resources. Therefore, it is very beneficial to contemplate time-cost trade-off. Moreover, workflow size and task size have a potential impact on the scheduling performance, which can be also further tackled in future research.

9. Future trends

As it can be observed from the various selected state-of-the-art scheduling techniques in this study that all issues (challenges) cannot be resolved using a single algorithm. For example, one algorithm may focus on energy, cost, time, and QoS parameters while these parameters are totally ignored by another which regards different parameters, such as resource utilization, availability, response time, and scalability, etc. These limitations should be triggered in future research, along with exploiting new opportunities arising from recent developments in the field [286–288]. Other several limitations of existing techniques can be overcome by devoting further research efforts into the following aspects:

- *Priority of users:* Cloud is ultimately a business model; thus, during the execution of submitted applications, the prioritization of cloud consumers should be taken into consideration.
- *Green computing:* Energy-aware task scheduling needs extensive research so that computing resources can be used more user- and environment-friendly by reducing the use of contaminated materials.
- *Resource controlling:* Key mechanisms, such as monitoring task migration, VM migration, memory or CPU utilization, etc., should be handled in a more controlled manner.
- *Workload prediction:* There is a need for more effective workload estimation techniques to predict the scale of upcoming workload, thereby increasing both the throughput and resource utilization.
- *Network Bandwidth:* Generally, network bandwidth has not received enough attention in the majority of current techniques, although disregarding it might cause communication delay, data loss, general network failure, etc.
- *Fog computing:* Traditional elastic cloud suffers from issues regarding security and delays which can be solved based on the new trend of fog computing which provides a higher level of heterogeneity and decentralization.
- *Failure prediction:* Resource failures including resource missing, storage failure, network failure, hardware failure, software failure, computing failure, database failure, overflow, underflow, and timeout can be predicted using diverse ML techniques.
- *Failure management:* The features regarding the management of task migration and failure have been tackled by a few scheduling algorithms; therefore, future research should address those features for maintaining the availability and constancy of the system.
- *IoT:* Managing the IoT devices and multimedia contents are critical recent trends on task scheduling in cloud.
- *Next generation computing:* Nano-computing-based/Quantum, non-traditional architecture is an attractive environment that should be involved in the next generation cloud.

10. Conclusions and recommendations

In general, solving a given scheduling problem using a meta-heuristic algorithm is very similar to launching a robot in a huge maze. The speed at which the robot locates the exit depends extremely on its search ability (vision) and decision ability (intelligence). Unfortunately, the optimum solution is not always what we get, especially with the existence of large and complex regions. Thus, the balance between intensification and diversification should be pursued in order to further enhance the quality of scheduling solutions. To this end and in order for the audience of this paper to deeply understand the meta-heuristic techniques delineated herein from a scheduling perspective in clouds, traditional and heuristic scheduling approaches were first introduced in order to differentiate them from their successive competitors, meta-heuristics. Then, a novel taxonomy was depicted to comprehensively and systematically categorize various prominent meta-heuristic scheduling approaches in the cloud, in terms of the nature of the scheduling problem, the primary objective of scheduling, the task-resource mapping scheme, and the scheduling constraint, taking into consideration the strengths and shortcomings of each approach. Based on the comparative analytics of the selected studies, it should be noted that PSO, GA, and ACO algorithms have been used by researchers to perform most of the work. Moreover, makespan, monetary cost, and resource utilization are the most active research areas. From another perspective, single objective scheduling, in general, reliability, scalability, and throughput objectives, in particular, budget and priority constraints, and AI-based and prediction-based mapping schemes have gained less researchers' attention; hence, they urgently need to be into the thick of future research.

Moreover, this study mainly aimed to vividly promote a breadth of the basic concepts of meta-heuristic task scheduling techniques in the cloud computing area, that should provide researchers and practitioners with tips on how to find out the nature of their scheduling problem, determine their primary QoS parameters, discover the appropriate task-resource mapping scheme, and specify the scheduling constraints most suitable for the problem, without breaching the SLA, for the potential of developing a novel scheduling algorithm. At the end, some simulation tools used in the field of cloud computing for implementing and testing new algorithms are briefly discussed and compared.

Succinctly, as of the end of this study, some concerns are posed. We do not say, however, that our study addressed all of these concerns and responds to them. Rather, we managed closely to help bridge the gap between meta-heuristic scheduling algorithms and their application to cloud task scheduling, which might help pave the way for further enhancements to the cloud service. This study also put forward various open research issues, along with future contemporary trends. An exciting area for future research involves selecting as the most appropriate resource (VM) as possible for incoming tasks/applications in order to optimize the process of task scheduling. For example, integration with a scalable architecture in accordance with user-defined QoS requirements can be suggested as a potential serious solution to this issue. Also, an autonomic QoS-based scheduling framework can be developed to help industrialists and researchers be more aware of the area of resource scheduling and optimization, especially with the advent of the era of Big Data and the IoT.

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.

CRediT authorship contribution statement

Essam H. Houssein: Supervision, Project administration, Conceptualization, Methodology, Formal analysis, Writing - review & editing. **Ahmed G. Gad:** Conceptualization, Methodology, Formal analysis, Visualization, Resources, Writing - original draft, Writing - review & editing. **Yaser M. Wazery:** Methodology, Formal analysis, Writing - review & editing. **Ponnuthurai Nagaratnam Suganthan:** Supervision, Writing - review & editing.

References

[1] M. Pinedo, K. Hadavi, Scheduling: theory, algorithms and systems development, in: Operations Research Proceedings 1991, Springer, 1992, pp. 35–42.

[2] A. Allahverdi, C. Ng, T.E. Cheng, M.Y. Kovalyov, A survey of scheduling problems with setup times or costs, Eur J Oper Res 187 (3) (2008) 985–1032.

[3] S.M. Johnson, Optimal two-and three-stage production schedules with setup times included, Naval research logistics quarterly 1 (1) (1954) 61–68.

[4] G.F. Pfister, In search of clusters, Prentice-Hall, Inc., 1998.

[5] I. Foster, The grid: a new infrastructure for 21st century science, Grid computing: making the global infrastructure a reality (2003) 51–63.

[6] C. Weinhardt, A. Anandasivam, B. Blau, N. Borissov, T. Meinl, W. Michalk, J. Stößer, Cloud computing–a classification, business models, and research directions, Business & Information Systems Engineering 1 (5) (2009) 391–399.

[7] K.R. Babu, P. Samuel, Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud, in: Innovations in bio-inspired computing and applications, Springer, 2016, pp. 67–78.

[8] M.R. Garey, D.S. Johnson, A guide to the theory of np-completeness, Computers and intractability (1979) 641–650.

[9] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, Eur J Oper Res 47 (1) (1990) 65–74.

[10] J.Y. Leung, Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC press, 2004.

[11] A. Allahverdi, The third comprehensive survey on scheduling problems with setup times/costs, Eur J Oper Res 246 (2) (2015) 345–378.

[12] T. Morton, D.W. Pentico, Heuristic Scheduling Systems: With Applications to Production Systems and Project Management, 3, John Wiley & Sons, 1993.

[13] D.C. Bissoli, W.A. Altoe, G.R. Mauri, A.R. Amaral, A simulated annealing meta-heuristic for the bi-objective flexible job shop scheduling problem, in: 2018 International Conference on Research in Intelligent and Computing in Engineering (RICE), IEEE, 2018, pp. 1–6.

[14] G. Gong, R. Chiong, Q. Deng, X. Gong, A hybrid artificial bee colony algorithm for flexible job shop scheduling with worker flexibility, Int. J. Prod. Res. (2019) 1–15.

[15] R. Zarrouk, I.E. Bennour, A. Jemai, A two-level particle swarm optimization algorithm for the flexible job shop scheduling problem, Swarm Intell. (2019) 1–24.

[16] N. Sadashiv, S.D. Kumar, Cluster, grid and cloud computing: A detailed comparison, in: 2011 6th International Conference on Computer Science & Education (ICCSE), IEEE, 2011, pp. 477–482.

[17] D. Garg, P. Kumar, A survey on metaheuristic approaches and its evaluation for load balancing in cloud computing, in: International Conference on Advanced Informatics for Computing Research, Springer, 2018, pp. 585–599.

[18] M. Kalra, S. Singh, A review of metaheuristic scheduling techniques in cloud computing, Egyptian informatics journal 16 (3) (2015) 275–295.

[19] N. Kaur, A. Chhabra, Analytical review of three latest nature inspired algorithms for scheduling in clouds, in: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, 2016, pp. 3296–3300.

[20] C.-W. Tsai, J.J. Rodrigues, Metaheuristic scheduling for cloud: a survey, IEEE Syst. J. 8 (1) (2013) 279–291.

[21] C. Nandhakumar, K. Ranjithprabhu, Heuristic and meta-heuristic workflow scheduling algorithms in multi-cloud environmentsa survey, in: 2015 International Conference on Advanced Computing and Communication Systems, IEEE, 2015, pp. 1–5.

[22] R. Kapur, Review of nature inspired algorithms in cloud computing, in: International Conference on Computing, Communication & Automation, IEEE, 2015, pp. 589–594.

[23] S. Shishira, A. Kandasamy, K. Chandrasekaran, Survey on meta heuristic optimization techniques in cloud computing, in: 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2016, pp. 1434–1440.

[24] P. Singh, M. Dutta, N. Aggarwal, A review of task scheduling based on meta-heuristics approach in cloud computing, Knowl Inf Syst 52 (1) (2017) 1–51.

[25] S.S. Chauhan, E.S. Pilli, R. Joshi, G. Singh, M. Govil, Brokering in interconnected cloud computing environments: a survey, J Parallel Distrib Comput 133 (2019) 193–209.

[26] G.K. Brar, A. Chhabra, Meta-heuristics based load balancing algorithms in grid and clouds-a review, in: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), IEEE, 2016, pp. 2938–2943.

[27] M. Rana, S. Bilgaiyan, U. Kar, A study on load balancing in cloud computing environment using evolutionary and swarm based algorithms, in: 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), IEEE, 2014, pp. 245–250.

[28] M. Masdari, F. Salehi, M. Jalali, M. Bidaki, A survey of PSO-based scheduling algorithms in cloud computing, Journal of Network and Systems Management 25 (1) (2017) 122–158.

[29] A. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in cloud computing: aliterature survey, Future Generation Computer Systems 91 (2019) 407–415.

[30] S. Bhosale, M. Parmar, D. Ambawade, A taxonomy and survey of manifold resource allocation techniques of iaas in cloud computing, in: International Conference on Sustainable Communication Networks and Application, Springer, 2019, pp. 191–202.

[31] H. Singh, S. Tyagi, P. Kumar, Scheduling in Cloud Computing Environment Using Metaheuristic Techniques: A Survey, in: Emerging Technology in Modelling and Graphics, Springer, 2020, pp. 753–763.

[32] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.R. Kan, Optimization and approximation in deterministic sequencing and scheduling: A survey, in: Annals of discrete mathematics, 5, Elsevier, 1979, pp. 287–326.

[33] A. Thakur, M.S. Goraya, A taxonomic survey on load balancing in cloud, Journal of Network and Computer Applications 98 (2017) 43–57.

[34] S. Singh, I. Chana, A survey on resource scheduling in cloud computing: issues and challenges, Journal of grid computing 14 (2) (2016) 217–264.

[35] M.J. Usman, A.S. Ismail, G. Abdul-Salaam, H. Chizari, O. Kaiwartya, A.Y. Gital, M. Abdullahi, A. Aliyu, S.I. Dishing, Energy-efficient nature-inspired techniques in cloud computing datacenters, Telecommun Syst 71 (2) (2019) 275–302.

[36] R. Jain, N. Sharma, P. Jain, A systematic analysis of nature inspired workflow scheduling algorithm in heterogeneous cloud environment, in: 2017 International Conference on Intelligent Communication and Computational Techniques (ICCT), IEEE, 2017, pp. 242–247.

[37] D. Tiwari, S. Singh, S. Sharma, Theoretical analysis of bio-inspired load balancing approach in cloud computing environment, International Journal of Database Theory and Application 10 (11) (2017) 15–26.

[38] B. Balusamy, J. Sridhar, D. Dhamodaran, P.V. Krishna, Bio-inspired algorithms for cloud computing: a review, Int. J. Innovative Comput. Appl. 6 (3–4) (2015) 181–202.

[39] S. Kaur, P. Bagga, R. Hans, H. Kaur, Quality of service (qos) aware workflow scheduling (wfs) in cloud computing: a systematic review, Arabian Journal for Science and Engineering 44 (4) (2019) 2867–2897.

[40] S. Potluri, K.S. Rao, Quality of service based task scheduling algorithms in cloud computing, International Journal of Electrical and Computer Engineering 7 (2) (2017) 1088.

[41] E.N. Alkhanak, S.P. Lee, R. Rezaei, R.M. Parizi, Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: a review, classifications, and open issues, Journal of Systems and Software 113 (2016) 1–26.

[42] F. Wu, Q. Wu, Y. Tan, Workflow scheduling in cloud: a survey, J Supercomput 71 (9) (2015) 3373–3418.

[43] R.V. Lopes, D. Menascé, A taxonomy of job scheduling on distributed computing systems, IEEE Trans. Parallel Distrib. Syst. 27 (12) (2016) 3412–3428.

[44] S. Imai, S. Patterson, C.A. Varela, Uncertainty-aware elastic virtual machine scheduling for stream processing systems, in: 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), IEEE, 2018, pp. 62–71.

[45] S.H.H. Madni, M.S.A. Latiff, Y. Coulibaly, S.M. Abdulhamid, An appraisal of meta-heuristic resource allocation techniques for iaas cloud, Indian Journal of Science and Technology 9 (4) (2016) 1–14.

[46] L. Thai, B. Varghese, A. Barker, A survey and taxonomy of resource optimisation for executing bag-of-task applications on public clouds, Future Generation Computer Systems 82 (2018) 1–11.

[47] S.H.H. Madni, M.S.A. Latiff, Y. Coulibaly, et al., Recent advancements in resource allocation techniques for cloud computing environment: a systematic review, Cluster Comput 20 (3) (2017) 2489–2533.

[48] S. Singh, I. Chana, Cloud resource provisioning: survey, status and future research directions, Knowl Inf Syst 49 (3) (2016) 1005–1069.

[49] B. Javadi, J. Abawajy, R.O. Sinnott, Hybrid cloud resource provisioning policy in the presence of resource failures, in: 4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, IEEE, 2012, pp. 10–17.

[50] K. Vukojevic-Haupt, F. Haupt, F. Leymann, On-demand provisioning of workflow middleware and services into the cloud: an overview, Computing 99 (2) (2017) 147–162.

[51] S. Khatua, P.K. Sur, R.K. Das, N. Mukherjee, Heuristic-based resource reservation strategies for public cloud, IEEE Trans. Cloud Comput. 4 (4) (2014) 392–401.

[52] B. Mikavica, A. Kostić-Ljubisavljević, Pricing and bidding strategies for cloud spot block instances, in: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), IEEE, 2018, pp. 0384–0389.

[53] S. Singh, I. Chana, Resource provisioning and scheduling in clouds: qos perspective, J Supercomput 72 (3) (2016) 926–960.

[54] Y. Shi, Z. Chen, W. Quan, M. Wen, A performance study of static task scheduling heuristics on cloud-scale acceleration architecture, in: Proceedings of the 2019 5th International Conference on Computing and Data Engineering, ACM, 2019, pp. 81–85.

[55] J. Li, T. Ma, M. Tang, W. Shen, Y. Jin, Improved fifo scheduling algorithm based on fuzzy clustering in cloud computing, Information 8 (1) (2017) 25.

[56] T. Nazar, N. Javaid, M. Waheed, A. Fatima, H. Bano, N. Ahmed, Modified shortest job first for load balancing in cloud-fog computing, in: International Conference on Broadband and Wireless Computing, Communication and Applications, Springer, 2018, pp. 63–76.

[57] D.C. Devi, V.R. Uthariaraj, Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks, The scientific world journal 2016 (2016).

[58] A.M.R. Mazumder, K.A. Uddin, N. Arbe, L. Jahan, M. Whaiduzzaman, Dynamic task scheduling algorithms in cloud computing, in: 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA), IEEE, 2019, pp. 1280–1286.

[59] S. Ghosh, C. Banerjee, Dynamic time quantum priority based round robin for load balancing in cloud environment, in: 2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN), IEEE, 2018, pp. 33–37.

[60] I. Gupta, M.S. Kumar, P.K. Jana, Task duplication-based workflow scheduling for heterogeneous cloud environment, in: 2016 Ninth International Conference on Contemporary Computing (IC3), IEEE, 2016, pp. 1–7.

[61] Y. Samadi, M. Zbakh, C. Tadonki, E-heft: enhancement heterogeneous earliest finish time algorithm for task scheduling based on load balancing in cloud computing, in: 2018 International Conference on High Performance Computing & Simulation (HPCS), IEEE, 2018, pp. 601–609.

[62] X. Ren, R. Lin, H. Zou, A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast, in: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, IEEE, 2011, pp. 220–224.

[63] M. Diallo, A. Quintero, S. Pierre, An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers, IEEE Trans. Cloud Comput. (2019).

[64] B. Jana, M. Chakraborty, T. Mandal, A task scheduling technique based on particle swarm optimization algorithm in cloud Environment, in: Soft Computing: Theories and Applications, Springer, 2019, pp. 525–536.

[65] N. Mansouri, B.M.H. Zade, M.M. Javidi, Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory, Computers & Industrial Engineering 130 (2019) 597–633.

[66] B. Li, L. Niu, X. Huang, H. Wu, Y. Pei, Minimum completion time offloading algorithm for mobile edge computing, in: 2018 IEEE 4th International Conference on Computer and Communications (ICCC), IEEE, 2018, pp. 1929–1933.

[67] H. Kasahara, A. Itoh, H. Tanaka, K. Itoh, A parallel optimization algorithm for minimum execution-time multiprocessor scheduling problem, Systems and computers in Japan 23 (13) (1992) 54–65.

[68] J. So, H. Byun, Load-balanced opportunistic routing for duty-cycled wireless sensor networks, IEEE Trans. Mob. Comput. 16 (7) (2016) 1940–1955.

[69] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, M. Naeem, Min-min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings, in: International Conference on Broadband and Wireless Computing, Communication and Applications, Springer, 2018, pp. 15–27.

[70] T.C. Hung, L.N. Hieu, P.T. Hy, N.X. Phi, Mmsia: Improved max-min scheduling algorithm for load balancing on cloud computing, in: Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, ACM, 2019, pp. 60–64.

[71] M.R. Belgaum, S. Soomro, Z. Alansari, M. Alam, S. Musa, M.M. Su'ud, Load balancing with preemptive and non-preemptive task scheduling in cloud computing, in: 2017 IEEE 3rd International Conference on Engineering Technologies and Social Sciences (ICETSS), IEEE, 2017, pp. 1–5.

[72] A. Kaleeswaran, V. Ramasamy, P. Vivekanandan, Dynamic scheduling of data using genetic algorithm in cloud computing, International Journal of Advances in Engineering & Technology 5 (2) (2013) 327.

[73] S. Patel, U. Bhoi, Priority based job scheduling techniques in cloud computing: a systematic review, International journal of scientific & technology research 2 (11) (2013) 147–152.

[74] T.L. Casavant, J.G. Kuhl, A taxonomy of scheduling in general-purpose distributed computing systems, IEEE Trans. Software Eng. 14 (2) (1988) 141–154.

[75] K.R. Baker, Introduction to sequencing and scheduling, John Wiley & Sons, 1974.

[76] A. Hatchuel, D. Saidi-Kabeche, J. Sardas, Towards a new planning and scheduling approach for multistage production systems, Int. J. Prod. Res. 35 (3) (1997) 867–886.

[77] E.L. Lawler, J.K. Lenstra, A.R. Kan, Recent developments in deterministic sequencing and scheduling: a survey, in: Deterministic and Stochastic Scheduling, Springer, 1982, pp. 35–73.

[78] S.H.H. Madni, M.S.A. Latiff, M. Abdullahi, S.M. Abdulhamid, M.J. Usman, Performance comparison of heuristic algorithms for task scheduling in iaas cloud computing environment, PLoS ONE 12 (5) (2017).

[79] A. Brandwajn, T. Begin, First-come-first-served queues with multiple servers and customer classes, Performance Evaluation 130 (2019) 51–63.

[80] M. Waheed, N. Javaid, A. Fatima, T. Nazar, K. Tehreem, K. Ansar, Shortest job first load balancing algorithm for efficient resource management in cloud, in: International Conference on Broadband and Wireless Computing, Communication and Applications, Springer, 2018, pp. 49–62.

[81] T. Balharith, F. Alhaidari, Round robin scheduling algorithm in cpu and cloud computing: A review, in: 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), IEEE, 2019, pp. 1–7.

[82] H. Krishnaveni, V.S.J. Prakash, Execution Time Based Sufferage Algorithm for Static Task Scheduling in Cloud, in: Advances in Big Data and Cloud Computing, Springer, 2019, pp. 61–70.

[83] M.A. Alworafi, A. Dhari, A.A. Al-Hashmi, A.B. Darem, et al., An improved sjf scheduling algorithm in cloud computing environment, in: 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), IEEE, 2016, pp. 208–212.

[84] S. Seth, N. Singh, Dynamic heterogeneous shortest job first (dhsjf): a task scheduling approach for heterogeneous cloud computing systems, International Journal of Information Technology 11 (4) (2019) 653–657.

[85] S. Elmougy, S. Sarhan, M. Joundy, A novel hybrid of shortest job first and round robin with dynamic variable quantum time task scheduling technique, Journal of Cloud Computing 6 (1) (2017) 1–12.

[86] M.A. Alworafi, A. Dhari, S.A. El-Booz, A.A. Nasr, A. Arpitha, S. Mallappa, An enhanced task scheduling in cloud computing based on hybrid approach, in: Data Analytics and Learning, Springer, 2019, pp. 11–25.

[87] H.S. Caranto, W.C.L. Olivete, J.V.D. Fernandez, C.A.R. Cabiara, R.B.M. Baquirin, E.F.G. Bayani, R.J.D. Fronda, Integrating user-defined priority tasks in a shortest job first round robin (sjfrr) scheduling algorithm, in: Proceedings of 2020 the 6th International Conference on Computing and Data Engineering, 2020, pp. 9–13.

[88] H. Chen, F. Wang, N. Helian, G. Akanmu, User-priority guided min-min scheduling algorithm for load balancing in cloud computing, in: 2013 national conference on parallel computing technologies (PARCOMPTECH), IEEE, 2013, pp. 1–8.

[89] D.G. Amalarethinam, S. Kavitha, Rescheduling enhanced min-min (remm) algorithm for meta-task scheduling in cloud computing, in: International Conference on Intelligent Data Communication Technologies and Internet of Things, Springer, 2018, pp. 895–902.

[90] Y. Mao, X. Chen, X. Li, Max–min task scheduling algorithm for load balance in cloud computing, in: Proceedings of International Conference on Computer Science and Information Technology, Springer, 2014, pp. 457–465.

[91] A.S. Karuppan, S.M. Kumari, S. Sruthi, A priority-based max-min scheduling algorithm for cloud environment using fuzzy approach, in: International Conference on Computer Networks and Communication Technologies, Springer, 2019, pp. 819–828.

[92] F. Saeed, N. Javaid, M. Zubair, M. Ismail, M. Zakria, M.H. Ashraf, M.B. Kamal, Load balancing on cloud analyst using first come first serve scheduling algorithm, in: International Conference on Intelligent Networking and Collaborative Systems, Springer, 2018, pp. 463–472.

[93] K. Dubey, M. Kumar, S. Sharma, Modified heft algorithm for task scheduling in cloud environment, Procedia Comput Sci 125 (2018) 725–732.

[94] B.L. Pan, Y.P. Wang, H.X. Li, J. Qian, Task scheduling and resource allocation of cloud computing based on qos, in: Advanced Materials Research, 915, Trans Tech Publ, 2014, pp. 1382–1385.

[95] X. Zhou, G. Zhang, J. Sun, J. Zhou, T. Wei, S. Hu, Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based heft, Future Generation Computer Systems 93 (2019) 278–289.

[96] R. Garg, A.K. Singh, Multi-objective workflow grid scheduling using ϵ-fuzzy dominance sort based discrete particle swarm optimization, J Supercomput 68 (2) (2014) 709–732.

[97] R. Garg, A.K. Singh, Multi-objective workflow grid scheduling based on discrete particle swarm optimization, in: International Conference on Swarm, Evolutionary, and Memetic Computing, Springer, 2011, pp. 183–190.
[98] J. Yu, M. Kirley, R. Buyya, Multi-objective planning for workflow execution on grids, in: 2007 8th IEEE/ACM International Conference on Grid Computing, IEEE, 2007, pp. 10–17.
[99] J.J. Durillo, R. Prodan, Multi-objective workflow scheduling in amazon ec2, Cluster Comput 17 (2) (2014) 169–189.
[100] J.J. Durillo, A.J. Nebro, Jmetal: a java framework for multi-objective optimization, Adv. Eng. Software 42 (10) (2011) 760–771.
[101] Z. Tong, X. Deng, H. Chen, J. Mei, H. Liu, Ql-heft: a novel machine learning scheduling scheme base on cloud computing environment, Neural Computing and Applications (2019) 1–18.
[102] H. Topcuoglu, S. Hariri, M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, IEEE Trans. Parallel Distrib. Syst. 13 (3) (2002) 260–274.
[103] J. Prassanna, N. Venkataraman, Threshold based multi-objective memetic optimized round robin scheduling for resource efficient load balancing in cloud, Mobile Networks and Applications 24 (4) (2019) 1214–1225.
[104] F.-H. Tseng, X. Wang, L.-D. Chou, H.-C. Chao, V.C. Leung, Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm, IEEE Syst. J. 12 (2) (2017) 1688–1699.
[105] L.-D. Chou, H.-F. Chen, F.-H. Tseng, H.-C. Chao, Y.-J. Chang, Dpra: dynamic power-saving resource allocation for cloud data center using particle swarm optimization, IEEE Syst. J. 12 (2) (2016) 1554–1565.
[106] N.A. Mehdi, A. Mamat, A. Amer, Z.T. Abdul-Mehdi, Minimum completion time for power-aware scheduling in cloud computing, in: 2011 Developments in E-systems Engineering, IEEE, 2011, pp. 484–489.
[107] D. Kliazovich, P. Bouvry, S.U. Khan, Greencloud: a packet-level simulator of energy-aware cloud computing data centers, J Supercomput 62 (3) (2012) 1263–1283.
[108] H. Krishnaveni, V.S. Janita, Completion time based sufferage algorithm for static task scheduling in cloud environment, International Journal of Pure and Applied Mathematics 119 (12) (2018) 13793–13797.
[109] J.E. Pecero, P. Bouvry, H.J.F. Huacuja, S.U. Khan, A multi-objective grasp algorithm for joint optimization of energy consumption and schedule length of precedence-constrained applications, in: 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, IEEE, 2011, pp. 510–517.
[110] F. Palmieri, U. Fiore, S. Ricciardi, A. Castiglione, Grasp-based resource re-optimization for effective big data access in federated clouds, Future Generation Computer Systems 54 (2016) 168–179.
[111] X. Chu, Y. Chen, Y. Tan, An anytime branch and bound algorithm for agile earth observation satellite onboard scheduling, Adv. Space Res. 60 (9) (2017) 2077–2090.
[112] H. Chen, Q. Liu, Q. Ai, A new heuristic scheduling strategy IBMM in cloud computing, in: 2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC), 1, IEEE, 2016, pp. 314–317.
[113] K. Sörensen, F. Glover, Metaheuristics, Encyclopedia of operations research and management science 62 (2013) 960–970.
[114] E.H. Houssein, A.G. Gad, Y.M. Wazery, Jaya algorithm and applications: a comprehensive review, Metaheuristics and Optimization in Computer and Electrical Engineering (2021) 3–24.
[115] J. Del Ser, E. Osaba, D. Molina, X.-S. Yang, S. Salcedo-Sanz, D. Camacho, S. Das, P.N. Suganthan, C.A.C. Coello, F. Herrera, Bio-inspired computation: where we stand and what's next, Swarm Evol Comput 48 (2019) 220–250.
[116] G. Beni, J. Wang, Swarm intelligence in cellular robotic systems, in: Robots and biological systems: towards a new bionics?, Springer, 1993, pp. 703–712.
[117] M. Dorigo, Optimization, learning and natural algorithms, PhD Thesis, Politecnico di Milano (1992).
[118] P. Lucic, D. Teodorovic, Transportation modeling: an artificial life approach, in: 14th IEEE International Conference on Tools with Artificial Intelligence, 2002.(ICTAI 2002). Proceedings., IEEE, 2002, pp. 216–223.
[119] S.D. Muller, J. Marchetto, S. Airaghi, P. Kournoutsakos, Optimization based on bacterial chemotaxis, IEEE Trans. Evol. Comput. 6 (1) (2002) 16–29.
[120] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, IEEE, 1995, pp. 39–43.
[121] G. Rjoub, J. Bentahar, Cloud task scheduling based on swarm intelligence and machine learning, in: 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud), IEEE, 2017, pp. 272–279.
[122] S. Asghari, N.J. Navimipour, Cloud service composition using an inverted ant colony optimisation algorithm, International Journal of Bio-Inspired Computation 13 (4) (2019) 257–268.
[123] B. Hajimirzaei, N.J. Navimipour, Intrusion detection for cloud computing using neural networks and artificial bee colony optimization algorithm, ICT Express 5 (1) (2019) 56–59.
[124] F. Gao, F.-X. Fei, H.-q. Tong, X.-j. Li, Bacterial foraging optimization oriented by atomized feature cloud model strategy, in: Proceedings of the 32nd Chinese Control Conference, IEEE, 2013, pp. 8032–8036.
[125] H. Ebrahimian, S. Barmayoon, M. Mohammadi, N. Ghadimi, The price prediction for the energy market based on a new method, Economic research-Ekonomska istraživanja 31 (1) (2018) 313–337.
[126] D.B. Fogel, Z. Michalewicz, T. Bäck, Handbook of evolutionary computation, Institute of Physics, 1997.
[127] J.H. Holland, Genetic algorithms and the optimal allocation of trials, SIAM J. Comput. 2 (2) (1973) 88–105.
[128] J.H. Holland, et al., Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence, MIT press, 1992.
[129] L.M. Khanli, S.N. Razavi, N.J. Navimipour, Lgr: The new genetic based scheduler for grid computing systems, in: 2008 International Conference on Computational Intelligence for Modelling Control & Automation, IEEE, 2008, pp. 639–644.
[130] F. Pop, C. Dobre, V. Cristea, Genetic algorithm for dag scheduling in grid environments, in: 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, IEEE, 2009, pp. 299–305.
[131] R. Storn, K. Price, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[132] M. Asafuddoula, T. Ray, R. Sarker, An adaptive hybrid differential evolution algorithm for single objective optimization, Appl Math Comput 231 (2014) 601–618.
[133] S. Kirkpatrick, Optimization by simulated annealing: quantitative studies, J Stat Phys 34 (5–6) (1984) 975–986.
[134] X.-S. Yang, Chapter 4 - simulated annealing, in: Nature-Inspired Optimization Algorithms, Elsevier, 2014, pp. 67–75.
[135] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76 (2) (2001) 60–68.
[136] R. Morsali, N. Ghadimi, M. Karimi, S. Mohajeryami, Solving a novel multiobjective placement problem of recloser and distributed generation sources in simultaneous mode by improved harmony search algorithm, Complexity 21 (1) (2015) 328–339.
[137] X.-S. Yang, Nature-inspired metaheuristic algorithms, Luniver press, 2010.
[138] S.-C. Chu, P.-W. Tsai, J.-S. Pan, Cat swarm optimization, in: Pacific Rim international conference on artificial intelligence, Springer, 2006, pp. 854–858.
[139] A. Cheraghalipour, M. Hajiaghaei-Keshteli, M.M. Paydar, Tree growth algorithm (tga): a novel approach for solving optimization problems, Eng Appl Artif Intell 72 (2018) 393–414.
[140] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Software 95 (2016) 51–67.
[141] G.-G. Wang, Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, Memetic Computing 10 (2) (2018) 151–164.
[142] M. Yazdani, F. Jolai, Lion optimization algorithm (loa): a nature-inspired metaheuristic algorithm, J. Comput. Des. Eng. 3 (1) (2016) 24–36.
[143] B. Alatas, Acroa: artificial chemical reaction optimization algorithm for global optimization, Expert Syst Appl 38 (10) (2011) 13170–13180.
[144] J. Pieter, E.D. de Jong, Evolutionary multi-agent systems, in: International Conference on Parallel Problem Solving from Nature, Springer, 2004, pp. 872–881.
[145] E. Rashedi, H. Nezamabadi-Pour, S. Saryazdi, Gsa: a gravitational search algorithm, Inf Sci (Ny) 179 (13) (2009) 2232–2248.
[146] F. Glover, Tabu searchpart i, ORSA Journal on computing 1 (3) (1989) 190–206.
[147] H.A. Abbass, R. Sarker, C. Newton, Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 2, IEEE, 2001, pp. 971–978.
[148] T. Mandal, S. Acharyya, Optimal task scheduling in cloud computing environment: Meta heuristic approaches, in: 2015 2nd International Conference on Electrical Information and Communication Technologies (EICT), IEEE, 2015, pp. 24–28.
[149] M.A. Elaziz, S. Xiong, K. Jayasena, L. Li, Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution, Knowl Based Syst 169 (2019) 39–52.
[150] F. Ramezani, J. Lu, F.K. Hussain, Task-based system load balancing in cloud computing using particle swarm optimization, Int J Parallel Program 42 (5) (2014) 739–754.
[151] F. Ramezani, J. Lu, J. Taheri, F.K. Hussain, Evolutionary algorithm-based multi--objective task scheduling optimization model in cloud environments, World Wide Web 18 (6) (2015) 1737–1757.
[152] L. Zuo, L. Shu, S. Dong, C. Zhu, T. Hara, A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing, IEEE Access 3 (2015) 2687–2699.
[153] H. He, G. Xu, S. Pang, Z. Zhao, Amts: adaptive multi-objective task scheduling strategy in cloud computing, China Commun. 13 (4) (2016) 162–171.
[154] S.H.H. Madni, M.S.A. Latiff, J. Ali, et al., Multi-objective-oriented cuckoo search optimization-based resource scheduling algorithm for clouds, Arabian Journal for Science and Engineering 44 (4) (2019) 3585–3602.
[155] F. Ramezani, J. Lu, F. Hussain, Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization, in: International Conference on Service-oriented Computing, Springer, 2013, pp. 237–251.
[156] L. Guo, S. Zhao, S. Shen, C. Jiang, Task scheduling optimization in cloud computing based on heuristic algorithm, Journal of networks 7 (3) (2012) 547.
[157] Y. Gao, G. Zhang, J. Lu, H.-M. Wee, Particle swarm optimization for bi-level pricing problems in supply chains, J. Global Optim. 51 (2) (2011) 245–254.
[158] H. Liu, A. Abraham, V. Snášel, S. McLoone, Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments, Inf Sci (Ny) 192 (2012) 228–243.
[159] F. Ramezani, J. Lu, F. Hussain, Task scheduling optimization in cloud computing applying multi-objective particle swarm optimization, in: International Conference on Service-oriented computing, Springer, 2013, pp. 237–251.
[160] R. Raju, R. Babukarthik, D. Chandramohan, P. Dhavachelvan, T. Vengattaraman, Minimizing the makespan using hybrid algorithm for cloud computing, in: 2013 3rd IEEE International Advance Computing Conference (IACC), IEEE, 2013, pp. 957–962.

[161] A. Khalili, S.M. Babamir, Makespan improvement of pso-based dynamic scheduling in cloud environment, in: 2015 23rd Iranian Conference on Electrical Engineering, IEEE, 2015, pp. 613–618.

[162] D. Gabi, A.S. Ismail, N.M. Dankolo, Minimized makespan based improved cat swarm optimization for efficient task scheduling in cloud datacenter, in: Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference, ACM, 2019, pp. 16–20.

[163] C. Malik, S. Jain, S. Randhawa, Resource scheduling in cloud using harmony search, in: 2016 International Conference on Inventive Computation Technologies (ICICT), 2, IEEE, 2016, pp. 1–6.

[164] M. Sharma, R. Garg, Higa: harmony-inspired genetic algorithm for rack-aware energy-efficient task scheduling in cloud data centers, Engineering Science and Technology, an International Journal (2019).

[165] J. Meena, M. Kumar, M. Vardhan, Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint, IEEE Access 4 (2016) 5065–5082.

[166] D. Chaudhary, B. Kumar, R. Khanna, Npso based cost optimization for load scheduling in cloud computing, in: International Symposium on Security in Computing and Communication, Springer, 2017, pp. 109–121.

[167] P. Han, C. Du, J. Chen, A dea based hybrid algorithm for bi-objective task scheduling in cloud computing, in: 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), IEEE, 2018, pp. 63–67.

[168] A.A. Nasr, N.A. El-Bahnasawy, G. Attiya, A. El-Sayed, Cost-effective algorithm for workflow scheduling in cloud computing under deadline constraint, Arabian Journal for Science and Engineering 44 (4) (2019) 3765–3780.

[169] H. Yuan, J. Bi, Profit-aware spatial task scheduling in distributed green clouds, in: 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), IEEE, 2019, pp. 421–426.

[170] Z. Wu, X. Liu, Z. Ni, D. Yuan, Y. Yang, A market-oriented hierarchical scheduling strategy in cloud workflow systems, J Supercomput 63 (1) (2013) 256–293.

[171] J. Thaman, M. Singh, Cost-effective task scheduling using hybrid approach in cloud, Int. J. Grid Util. Comput. 8 (3) (2017) 241–253.

[172] D. Chaudhary, B. Kumar, Cost optimized hybrid genetic-gravitational search algorithm for load scheduling in cloud computing, Appl Soft Comput 83 (2019) 105627.

[173] M.E. Frincu, C. Craciun, Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments, in: 2011 fourth IEEE international conference on utility and cloud computing, IEEE, 2011, pp. 267–274.

[174] H.R. Faragardi, R. Shojaee, N. Yazdani, Reliability-aware task allocation in distributed computing systems using hybrid simulated annealing and tabu search, in: 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems, IEEE, 2012, pp. 1088–1095.

[175] H.R. Faragardi, R. Shojaee, M.A. Keshtkar, H. Tabani, Optimal task allocation for maximizing reliability in distributed real-time systems, in: 2013 IEEE/ACIS 12th International Conference on Computer and Information Science (ICIS), IEEE, 2013, pp. 513–519.

[176] H. Cui, Y. Li, X. Liu, N. Ansari, Y. Liu, Cloud service reliability modelling and optimal task scheduling, IET Commun. 11 (2) (2017) 161–167.

[177] D. Gabi, A.S. Ismail, A. Zainal, Z. Zakaria, A. Al-Khasawneh, Cloud scalable multi-objective task scheduling algorithm for cloud computing using cat swarm optimization and simulated annealing, in: 2017 8th International Conference on Information Technology (ICIT), IEEE, 2017, pp. 599–604.

[178] D. Gabi, A. Zainal, A.S. Ismail, Z. Zakaria, Scalability-aware scheduling optimization algorithm for multi-objective cloud task scheduling problem, in: 2017 6th ICT International Student Project Conference (ICT-ISPC), IEEE, 2017, pp. 1–6.

[179] K. Pradeep, T.P. Jacob, A hybrid approach for task scheduling using the cuckoo and harmony search in cloud computing environment, Wireless Personal Communications 101 (4) (2018) 2287–2311.

[180] I. Strumberger, E. Tuba, N. Bacanin, M. Tuba, Dynamic tree growth algorithm for load scheduling in cloud environments, in: 2019 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2019, pp. 65–72.

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

[182] S. Yassa, R. Chelouah, H. Kadima, B. Granado, Multi-objective approach for energy-aware workflow scheduling in cloud computing environments, The Scientific World Journal 2013 (2013).

[183] F. Tao, Y. Feng, L. Zhang, T.W. Liao, Clps-ga: a case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling, Appl Soft Comput 19 (2014) 264–279.

[184] J. Meshkati, F. Safi-Esfahani, Energy-aware resource utilization based on particle swarm optimization and artificial bee colony algorithms in cloud computing, J Supercomput 75 (5) (2019) 2455–2496.

[185] A. Goyal, N.S. Chahal, Bio inspired approach for load balancing to reduce energy consumption in cloud data center, in: 2015 Communication, Control and Intelligent Systems (CCIS), IEEE, 2015, pp. 406–410.

[186] S.M. Abdulhamid, M.S.A. Latiff, G. Abdul-Salaam, S.H.H. Madni, Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm, PLoS ONE 11 (7) (2016).

[187] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, B. Luo, A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds, Future Generation Computer Systems 65 (2016) 140–152.

[188] Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao, J. Chen, Scheduling workflows with privacy protection constraints for big data applications on cloud, Future Generation Computer Systems (2018).

[189] J.A.J. Sujana, T. Revathi, T.S. Priya, K. Muneeswaran, Smart pso-based secured scheduling approaches for scientific workflows in cloud computing, Soft comput 23 (5) (2019) 1745–1765.

[190] M.R. Thanka, P.U. Maheswari, E.B. Edwin, An improved efficient: artificial bee colony algorithm for security and qos aware scheduling in cloud computing environment, Cluster Comput 22 (5) (2019) 10905–10913.

[191] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, A. Abraham, Hybrid job scheduling algorithm for cloud computing environment, in: Proceedings of the fifth international conference on innovations in bio-inspired computing and applications IBICA 2014, Springer, 2014, pp. 43–52.

[192] R. Kumari, A. Jain, An efficient resource utilization based integrated task scheduling algorithm, in: 2017 4th International Conference on Signal Processing and Integrated Networks (SPIN), IEEE, 2017, pp. 519–523.

[193] S. Rani, P. Suri, An efficient and scalable hybrid task scheduling approach for cloud environment, International Journal of Information Technology (2018) 1–7.

[194] X. Chen, L. Cheng, C. Liu, Q. Liu, J. Liu, Y. Mao, J. Murphy, A woa-based optimization approach for task scheduling in cloud computing systems, IEEE Syst. J. (2020).

[195] G. Shobana, M. Geetha, R. Suganthe, Nature inspired preemptive task scheduling for load balancing in cloud datacenter, in: International Conference on Information Communication and Embedded Systems (ICICES2014), IEEE, 2014, pp. 1–6.

[196] S.H.H. Madni, M.S.A. Latiff, J. Ali, et al., Hybrid gradient descent cuckoo search (hgdcs) algorithm for resource scheduling in iaas cloud computing environment, Cluster Comput 22 (1) (2019) 301–334.

[197] S. Abrishami, M. Naghibzadeh, D.H. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, Future Generation Computer Systems 29 (1) (2013) 158–169.

[198] D. Poola, S.K. Garg, R. Buyya, Y. Yang, K. Ramamohanarao, Robust scheduling of scientific workflows with deadline and budget constraints in clouds, in: 2014 IEEE 28th international conference on advanced information networking and applications, IEEE, 2014, pp. 858–865.

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

[200] M.A. Rodriguez, R. Buyya, Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds, IEEE Trans. Cloud Comput. 2 (2) (2014) 222–235.

[201] M.-Y. Cheng, D. Prayogo, Symbiotic organisms search: a new metaheuristic optimization algorithm, Computers & Structures 139 (2014) 98–112.

[202] X. Deng, D. Wu, J. Shen, J. He, Eco-aware online power management and load scheduling for green cloud datacenters, IEEE Syst. J. 10 (1) (2014) 78–87.

[203] J. Bi, H. Yuan, W. Tan, B.H. Li, Trs: temporal request scheduling with bounded delay assurance in a green cloud data center, Inf Sci (Ny) 360 (2016) 57–72.

[204] D. Chaudhary, B. Kumar, Linear improved gravitational search algorithm for load scheduling in cloud computing environment (LIGSA-C), International Journal of Computer Network and Information Security 10 (4) (2018) 38.

[205] Y.C. Lee, A.Y. Zomaya, Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling, in: 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, IEEE, 2009, pp. 92–99.

[206] M. Yue, A simple proof of the inequality $ffd(l) \leq 11/9opt(l) + 1, \forall l$ for the ffd bin-packing algorithm, Acta mathematicae applicatae sinica 7 (4) (1991) 321–331.

[207] T. Setzer, A. Stage, Decision support for virtual machine reassignments in enterprise data centers, in: 2010 IEEE/IFIP Network Operations and Management Symposium Workshops, IEEE, 2010, pp. 88–94.

[208] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.

[209] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.

[210] H. Arabnejad, J.G. Barbosa, List scheduling algorithm for heterogeneous systems by an optimistic cost table, IEEE Trans. Parallel Distrib. Syst. 25 (3) (2013) 682–694.

[211] Z. Wu, Z. Ni, L. Gu, X. Liu, A revised discrete particle swarm optimization for cloud workflow scheduling, in: 2010 International Conference on Computational Intelligence and Security, IEEE, 2010, pp. 184–188.

[212] W. Yonggui, H. Ruilian, Study on cloud computing task schedule strategy based on maco algorithm, Computer Measurement & Control 5 (2011).

[213] R.J. Priyadarsini, L. Arockiam, Pbcopso: a parallel optimization algorithm for task scheduling in cloud environment, Indian J Sci Technol 8 (2015) 6–10.

[214] D.B. LD, P.V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, Appl Soft Comput 13 (5) (2013) 2292–2303.

[215] J.G.D. Matos, C.K.D.M. Marques, C.H. Liberalino, Genetic and static algorithm for task scheduling in cloud computing, International Journal of Cloud Computing 8 (1) (2019) 1–19.

[216] B. Shirazi, A. Hurson, K. Kavi, Introduction to scheduling and load balancing, IEEE Computer Society (1995).

[217] F. Xhafa, A. Abraham, Meta-heuristics for grid scheduling problems, in: Metaheuristics for Scheduling in Distributed Computing Environments, Springer, 2008, pp. 1–37.

[218] M.R. Islam, M. Habiba, Dynamic scheduling approach for data-intensive cloud environment, in: 2012 International Conference on Cloud Computing Technologies, Applications and Management (ICCCTAM), IEEE, 2012, pp. 179–185.

[219] M. Rahman, R. Hassan, R. Ranjan, R. Buyya, Adaptive workflow scheduling for dynamic grid and cloud computing environment, Concurrency and Computation: Practice and Experience 25 (13) (2013) 1816–1842.
[220] H.B. Alla, S.B. Alla, A. Ezzati, A novel architecture for task scheduling based on dynamic queues and particle swarm optimization in cloud computing, in: 2016 2nd International Conference on Cloud Computing Technologies and Applications (CloudTech), IEEE, 2016, pp. 108–114.
[221] M.A. Haghighi, M. Maeen, M. Haghparast, An energy-efficient dynamic resource management approach based on clustering and meta-heuristic algorithms in cloud computing iaas platforms, Wireless Personal Communications 104 (4) (2019) 1367–1391.
[222] F. Hemasian-Etefagh, F. Safi-Esfahani, Dynamic scheduling applying new population grouping of whales meta-heuristic in cloud computing, J Supercomput 75 (10) (2019) 6386–6450.
[223] P.T. Endo, A.V. de Almeida Palhares, N.N. Pereira, G.E. Goncalves, D. Sadok, J. Kelner, B. Melander, J.-E. Mangs, Resource allocation for distributed cloud: concepts and research challenges, IEEE Netw 25 (4) (2011) 42–46.
[224] N. Kumar, P. Patel, Resource management using feed forward ann-pso in cloud computing environment, in: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, ACM, 2016, pp. 1–6.
[225] S. Negi, N. Panwar, K.S. Vaisla, M.M.S. Rauthan, Artificial neural network based load balancing in cloud environment, in: Advances in Data and Information Sciences, Springer, 2020, pp. 203–215.
[226] X. Gao, R. Liu, A. Kaushik, Hierarchical multi-agent optimization for resource allocation in cloud computing, arXiv preprint arXiv:2001.03929 (2020).
[227] M. Sharma, R. Garg, An artificial neural network based approach for energy efficient task scheduling in cloud data centers, Sustainable Computing: Informatics and Systems (2020) 100373.
[228] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Generation Computer Systems 28 (1) (2012) 155–162.
[229] R. Patel, D. Dahiya, Aggregation of cloud providers: A review of opportunities and challenges, in: International Conference on Computing, Communication & Automation, IEEE, 2015, pp. 620–626.
[230] H. Hu, H. Wang, A prediction-based aco algorithm to dynamic tasks scheduling in cloud environment, in: 2016 2nd IEEE International Conference on Computer and Communications (ICCC), IEEE, 2016, pp. 2727–2732.
[231] V. Vashishth, A. Chhabra, A. Sood, A predictive approach to task scheduling for big data in cloud environments using classification algorithms, in: 2017 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence, IEEE, 2017, pp. 188–192.
[232] E. Gabaldon, S. Vila, F. Guirado, J.L. Lerida, J. Planes, Energy efficient scheduling on heterogeneous federated clusters using a fuzzy multi-objective meta-heuristic, in: 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), IEEE, 2017, pp. 1–6.
[233] W. Li, Y. Xia, M. Zhou, X. Sun, Q. Zhu, Fluctuation-aware and predictive workflow scheduling in cost-effective infrastructure-as-a-service clouds, IEEE Access 6 (2018) 61488–61502.
[234] B. Yagoubi, Y. Slimani, Task load balancing strategy for grid computing, Journal of Computer Science 3 (3) (2007) 186–194.
[235] B. Yagoubi, Y. Slimani, Dynamic load balancing strategy for grid computing, Transactions on Engineering, Computing and Technology 13 (2006) (2006) 260–265.
[236] M.R. Islam, M.T. Hasan, G. Ashaduzzaman, An architecture and a dynamic scheduling algorithm of grid for providing security for real-time data-intensive applications, Int. J. Network Manage. 21 (5) (2011) 402–413.
[237] K. Ranganathan, I. Foster, Decoupling computation and data scheduling in distributed data-intensive applications, in: Proceedings 11th IEEE International Symposium on High Performance Distributed Computing, IEEE, 2002, pp. 352–358.
[238] M. Wieczorek, R. Prodan, T. Fahringer, Scheduling of scientific workflows in the askalon grid environment, Acm Sigmod Record 34 (3) (2005) 56–62.
[239] J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, K. Kennedy, Task scheduling strategies for workflow-based applications in grids, in: CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005., 2, IEEE, 2005, pp. 759–767.
[240] S. Nesmachnow, B. Dorronsoro, J.E. Pecero, P. Bouvry, Energy-aware scheduling on multicore heterogeneous grid computing systems, Journal of grid computing 11 (4) (2013) 653–680.
[241] J.L. Berral, Í. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, J. Torres, Towards energy-aware scheduling in data centers using machine learning, in: Proceedings of the 1st International Conference on energy-Efficient Computing and Networking, 2010, pp. 215–224.
[242] W.M. Jones, W.B. Ligon, L.W. Pang, D. Stanzione, Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters, J Supercomput 34 (2) (2005) 135–163.
[243] V.K. Naik, C. Liu, L. Yang, J. Wagner, Online resource matching for heterogeneous grid environments, in: CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005., 2, IEEE, 2005, pp. 607–614.
[244] E. Gabaldon, F. Guirado, J.L. Lerida, J. Planes, Particle swarm optimization scheduling for energy saving in cluster computing heterogeneous environments, in: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), IEEE, 2016, pp. 321–325.
[245] H. Ma, H. Zhu, Z. Hu, W. Tang, P. Dong, Multi-valued collaborative qos prediction for cloud service via time series analysis, Future Generachen2016newtion Computer Systems 68 (2017) 275–288.
[246] Z. Zhu, G. Zhang, M. Li, X. Liu, Evolutionary multi-objective workflow scheduling in cloud, IEEE Trans. Parallel Distrib. Syst. 27 (5) (2015) 1344–1357.
[247] A.A. Visheratin, M. Melnik, D. Nasonov, Workflow scheduling algorithms for hard-deadline constrained cloud environments, Procedia Comput Sci 80 (2016) 2098–2106.
[248] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, J. Wen, Deadline-constrained cost optimization approaches for workflow scheduling in clouds, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017) 3401–3412.
[249] A.K. Maurya, A.K. Tripathi, Deadline-constrained algorithms for scheduling of bag-of-tasks and workflows in cloud computing environments, in: Proceedings of the 2nd International Conference on High Performance Compilation, Computing and Communications, ACM, 2018, pp. 6–10.
[250] A. Verma, S. Kaushal, Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud, in: 2014 Recent Advances in Engineering and Computational Sciences (RAECS), IEEE, 2014, pp. 1–6.
[251] S.T. Milan, L. Rajabion, A. Darwesh, M. Hosseinzadeh, N.J. Navimipour, Priority-based task scheduling method over cloudlet using a swarm intelligence algorithm, Cluster Comput (2019) 1–9.
[252] A. Verma, S. Kaushal, Budget constrained priority based genetic algorithm for workflow scheduling in cloud, in: Proceedings of the Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013), IET, 2013, pp. 216–222.
[253] X. Wang, B. Cao, C. Hou, L. Xiong, J. Fan, Scheduling budget constrained cloud workflows with particle swarm optimization, in: 2015 IEEE Conference on Collaboration and Internet Computing (CIC), IEEE, 2015, pp. 219–226.
[254] P. Guo, Z. Xue, Cost-effective fault-tolerant scheduling algorithm for real-time tasks in cloud systems, in: 2017 IEEE 17th International Conference on Communication Technology (ICCT), IEEE, 2017, pp. 1942–1946.
[255] S.M. Abdulhamid, M.S.A. Latiff, A checkpointed league championship algorithm-based cloud scheduling scheme with secure fault tolerance responsiveness, Appl Soft Comput 61 (2017) 670–680.
[256] S.M. Abdulhamid, M.S.A. Latiff, S.H.H. Madni, M. Abdullahi, Fault tolerance aware scheduling technique for cloud computing environment using dynamic clustering algorithm, Neural Computing and Applications 29 (1) (2018) 279–293.
[257] M. Mao, M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows, in: SC'11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis, IEEE, 2011, pp. 1–12.
[258] W. Zheng, R. Sakellariou, Budget-deadline constrained workflow planning for admission control in market-oriented environments, in: International Workshop on Grid Economics and Business Models, Springer, 2011, pp. 105–119.
[259] E.S. Alkayal, N.R. Jennings, M.F. Abulkhair, Efficient task scheduling multi-objective particle swarm optimization in cloud computing, in: 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), IEEE, 2016, pp. 17–24.
[260] Y. Dai, Y. Lou, X. Lu, A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-qos constraints in cloud computing, in: 2015 7th International Conference on Intelligent Human-Machine Systems and Cybernetics, 2, IEEE, 2015, pp. 428–431.
[261] M. Abdullahi, M.A. Ngadi, et al., Symbiotic organism search optimization based task scheduling in cloud computing environment, Future Generation Computer Systems 56 (2016) 640–650.
[262] Y. Changtian, Y. Jiong, Energy-aware genetic algorithms for task scheduling in cloud computing, in: 2012 Seventh ChinaGrid Annual Conference, IEEE, 2012, pp. 43–48.
[263] H.M. Fard, R. Prodan, T. Fahringer, Multi-objective list scheduling of workflow applications in distributed computing infrastructures, J Parallel Distrib Comput 74 (3) (2014) 2152–2165.
[264] L. Ramakrishnan, D.A. Reed, Performability modeling for scheduling and fault tolerance strategies for scientific workflows, in: Proceedings of the 17th International Symposium on High Performance Distributed Computing, 2008, pp. 23–34.
[265] H. Xu, B. Yang, W. Qi, E. Ahene, A multi-objective optimization approach to workflow scheduling in clouds considering fault recovery., KSII Transactions on Internet & Information Systems 10 (3) (2016).
[266] M.A. Tawfeek, A. El-Sisi, A.E. Keshk, F.A. Torkey, Cloud task scheduling based on ant colony optimization, in: 2013 8th International Conference on Computer Engineering & Systems (ICCES), IEEE, 2013, pp. 64–69.
[267] J. Gąsior, F. Seredyński, Multi-objective parallel machines scheduling for fault-tolerant cloud systems, in: International Conference on Algorithms and Architectures for Parallel Processing, Springer, 2013, pp. 247–256.
[268] R. Buyya, M. Murshed, GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Concurrency and computation: practice and experience 14 (13–15) (2002) 1175–1220.
[269] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A. De Rose, R. Buyya, Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and experience 41 (1) (2011) 23–50.
[270] B. Wickremasinghe, R.N. Calheiros, R. Buyya, Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications, in: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, IEEE, 2010, pp. 446–452.
[271] R.N. Calheiros, M.A. Netto, C.A. De Rose, R. Buyya, EMUSim: an integrated emulation and simulation environment for modeling, evaluation, and validation of performance of cloud computing applications, Software: Practice and Experience 43 (5) (2013) 595–612.

[272] S.K. Garg, R. Buyya, NetworkCloudSim: Modelling parallel applications in cloud simulations, in: 2011 Fourth IEEE International Conference on Utility and Cloud Computing, IEEE, 2011, pp. 105–113.
[273] W. Chen, E. Deelman, WorkflowSim: A toolkit for simulating scientific workflows in distributed environments, in: 2012 IEEE 8th International Conference on E-Science, IEEE, 2012, pp. 1–8.
[274] S. Ostermann, K. Plankensteiner, R. Prodan, T. Fahringer, Groudsim: an event-based simulation framework for computational grids and clouds, in: European Conference on Parallel Processing, Springer, 2010, pp. 305–313.
[275] A. Núñez, J.L. Vázquez-Poletti, A.C. Caminero, G.G. Castañé, J. Carretero, I.M. Llorente, Icancloud: a flexible and scalable cloud infrastructure simulator, Journal of Grid Computing 10 (1) (2012) 185–209.
[276] N. Alaei, F. Safi-Esfahani, Repro-active: a reactive–proactive scheduling method based on simulation in cloud computing, J Supercomput 74 (2) (2018) 801–829.
[277] C. Jianfang, C. Junjie, Z. Qingshan, An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm, Cybernetics and Information Technologies 14 (1) (2014) 25–39.
[278] E. Atashpaz-Gargari, C. Lucas, Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation, IEEE, 2007, pp. 4661–4667.
[279] X.-S. Yang, A.H. Gandomi, Bat algorithm: a novel approach for global engineering optimization, Eng Comput (Swansea) (2012).
[280] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, Future generation computer systems 97 (2019) 849–872.
[281] S. Mirjalili, S.M. Mirjalili, A. Lewis, Grey wolf optimizer, Adv. Eng. Software 69 (2014) 46–61.
[282] Z. Bayraktar, M. Komurcu, D.H. Werner, Wind driven optimization (wdo): A novel nature-inspired optimization algorithm and its application to electromagnetics, in: 2010 IEEE antennas and propagation society international symposium, IEEE, 2010, pp. 1–4.
[283] S.M.G. Kashikolaei, A.A.R. Hosseinabadi, B. Saemi, M.B. Shareh, A.K. Sangaiah, G.-B. Bian, An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm, J Supercomput (2019) 1–28.
[284] S. Raghavan, P. Sarwesh, C. Marimuthu, K. Chandrasekaran, Bat algorithm for scheduling workflow applications in cloud, in: 2015 International Conference on Electronic Design, Computer Networks & Automated Verification (EDCAV), IEEE, 2015, pp. 139–144.
[285] V. Khajehvand, H. Pedram, M. Zandieh, Sctts: scalable cost-time trade-off scheduling for workflow application in grids., KSII Transactions on Internet & Information Systems 7 (12) (2013).
[286] B. Varghese, R. Buyya, Next generation cloud computing: new trends and research directions, Future Generation Computer Systems 79 (2018) 849–861.
[287] K. Ujjwal, S. Garg, J. Hilton, J. Aryal, N. Forbes-Smith, Cloud computing in natural hazard modeling systems: current research trends and future directions, Int. J. Disaster Risk Reduct. (2019) 101188.
[288] N.K. Sehgal, P.C.P. Bhatt, J.M. Acken, Future trends in cloud computing, in: Cloud Computing with Security, Springer, 2020, pp. 235–259.