Comput Syst Sci & Eng (2019) 6: 357–376

International Journal of
Computer Systems
Science & Engineering

Application Layer Scheduling in Cloud: Fundamentals, Review and Research Directions

Vaibhav Pandey* **and Poonam Saini†**

Department of Computer Science & Engineering
Punjab Engineering College (Deemed to be University), Chandigarh

The cloud computing paradigm facilitates a finite pool of on-demand virtualized resources on a pay-per-use basis. For large-scale heterogeneous distributed systems like a cloud, *scheduling* is an essential component of resource management at the application layer as well as at the virtualization layer in order to deliver the optimal Quality of Services (QoS). The cloud scheduling, in general, is an NP-hard problem due to large solution space, thus, it is difficult to find an optimal solution within a reasonable time. In application layer scheduling, the tasks are mapped to logical resources (i.e., virtual machines), aiming to optimize one or more QoS parameters, and conforming to several constraints. Various algorithms have been proposed in the literature for application layer scheduling, where each of them is based on some fundamental design techniques like simple heuristics, meta-heuristics, and most recently hybrid heuristics. Although ample literature survey exists for cloud scheduling algorithms, none of them present their study exclusively for the application layer. In this survey paper, we present a study on task scheduling algorithms used only at the application layer of the cloud. We classify our study according to various fundamental techniques used in designing such scheduling algorithms. One of the main features of our study is that it covers numerous application type e.g., a set of independent tasks, simple workflow, scientific workflow, and MapReduce jobs. We also provide a comparative analysis of existing algorithms on various parameters like makespan, cost, resource utilization, etc. In the end, research directions for future work have been provided

Keywords: Application layer; Task scheduling; Cloud; Resource management

1. INTRODUCTION

The cloud computing technology provides a virtualized pool of on-demand resources and is based on a pay-per-use model [1] [2]. The concept has been perceived as a realization of utility computing where computing facilities are provided over the network similar to other utility services like water, electricity, telephone, etc. With an increase in on-demand resource allocation and utility-based pricing, cloud service providers can truly maximize the utilization of resources and thus minimize operating costs [3]. Also, cloud computing technology plays a major role in the success of various business enterprises as it eliminates the essential requirements to plan before provisioning resources. Hence, any business enterprise may start with a small virtual setup and scale the computing resources as per demand [4]. Based on location, accessibility, and administration, a cloud may be classified as public, private, hybrid or cloud federation [5] [6]. On the basis of provided services, a cloud computing architecture has three layers, namely, Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [7] [8] [9]. Further, with the advancement in cloud computing technology, various other computing services like storage, network infrastructure, security, and data analytics have emerged and classified under Anything as a Service (XaaS) [10].

A cloud comprises hardware resources like servers, switches, cooling and power infrastructure, etc., and software resources like user applications, virtual machine monitors (VMMs), operating system (OS) and cloud software [1][2]. The resources are organized in different layers for simple management and in order to deliver services to the users efficiently. Fig. 1 shows the layered architecture of the cloud [3]. The resource management

*pandeyvaibhav51@gmail.com
†poonamsaini@pec.ac.in

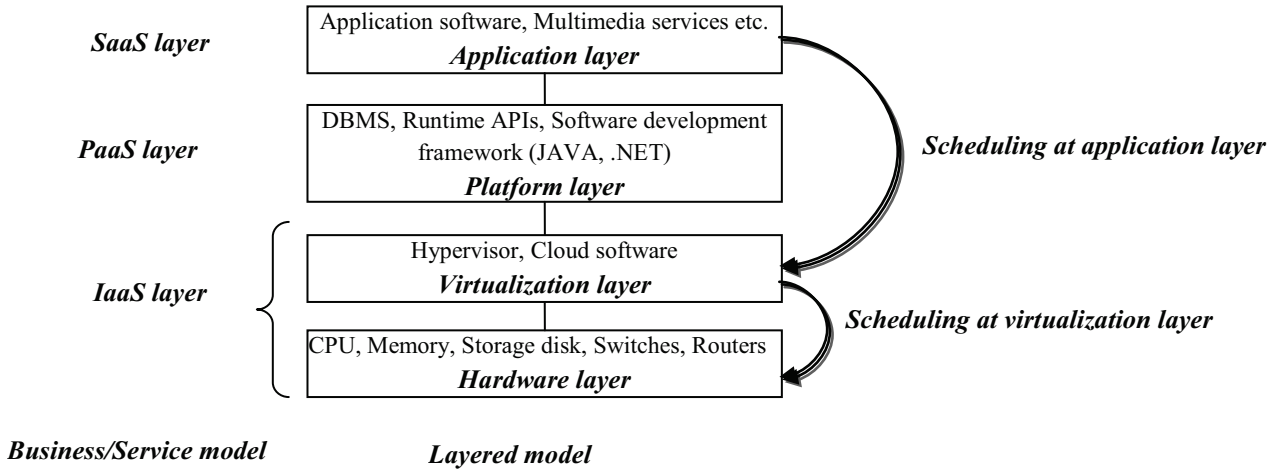

Figure 1 Layered architecture of cloud.

in cloud computing paradigm focuses on the organization and usages of these resources to optimize energy consumption, cost, execution time and other quality of service (QoS) parameters. It further includes the efficient establishment of networking infrastructure for better communication among servers within data centers, and management of efficient power and cooling infrastructure, etc. On the other hand, scheduling is just a tool of resource management at the application and virtualization layer of cloud. As the problem of scheduling is generally taken into consideration at two layers of the cloud service stack, the cloud scheduling problem can be classified into *scheduling at the application layer*[1] and *scheduling at virtualization layer*[2] [3] as illustrated in Fig. 1. The scheduling at application layer allocates various independent tasks, user applications, workflow tasks, etc. to virtual machines. Whereas, at the *virtualization layer*, scheduling focuses on the process of mapping virtual machines on physical machines (i.e., servers)usually considering parameters such as energy conservation, resource utilization, and optimal load balance. In this work, we study only application layer scheduling algorithms. In the view of the above discussion, the term resource management refers to the overall management all resources available in a data center to optimize the QoS parameters whereas, at application and virtualization layer, it is handled by scheduling in order to assign shared resources from the lower layer to upper layer tasks.

Traditional scheduling schemes that are used in multiprocessor, cluster and Grid environment fails to work in cloud computing systems due to its specific characteristics like virtualization, elasticity, and pay-per-use model. Therefore, to overcome such limitations, various *heuristic*, *meta-heuristic* and *hybrid-heuristic* techniques have been adopted to design scheduling algorithms at the application layer. The heuristic techniques are problem-dependent, thus, cannot be applicable to every problem. On the other hand, the meta-heuristics techniques are problem-independent and can be applied to a wide variety of situations. The hybrid-heuristic technique merges the previous two to take advantage in terms of improved efficiency of an individual scheme. The task scheduling algorithms studied in the paper have been classified on the basis of above-mentioned fundamental techniques.

An adequate number of literature survey works [4], [5], [14], [15], [6]–[13] have attempted to study various cloud scheduling algorithms at multiple layers. Some of them discussed the cloud schedulers which are designed only for a specific QoS objective. Whereas, some of them studied the cloud scheduler designed specifically for a particular application. Authors in [5]–[7], [11] present a comprehensive study of energy-efficient cloud schedulers at application and virtualization layer. Authors in [10][9] discussed only workflow schedulers at various layers. Although the existing surveys are comprehensive, none of them focused exclusively at the application layer for scheduling of different categories of applications. In this survey paper, we include different kind of applications e.g., a set of independent tasks, a simple workflow, scientific workflow, MapReduce jobs, etc. and studied various cloud scheduler proposed for them. We classify our study on the basis of the fundamental technique used to design such schedulers.

The rest of the paper is organized as follows. Section 2 introduces the generic and cloud-based scheduling problem. Section 3 describes the application layer cloud scheduling in detail with various QoS requirements and characteristics of scheduling algorithms. Section 4 presents the study of the different cloud scheduling algorithms at the application layer classified according to the fundamental technique used to design it. The open challenges and current research trends have been presented in section 5, followed by the conclusion in section 6.

2. THE SCHEDULING PROBLEM: GENERIC AND CLOUD-BASED

The scheduling problem in computer science has evolved from single processor to multiprocessor, and then to large-scale multi-computer systems e.g., Grid, Cluster, and Cloud [16]. The cloud scheduling problem is similar to, multi-processor scheduling problem where one has to find an optimal solution for scheduling a given set of tasks $T = \{t_1, t_2, \ldots\ldots t_n\}$ to a given set of machines $M = \{m_1, m_2 \ldots\ldots m_m\}$ with one or more performance objectives to be optimized subject to some

[1] Scheduling at application layer is usually termed as task scheduling

[2] Scheduling at virtualization layer is generally termed as VM scheduling or resource provisioning

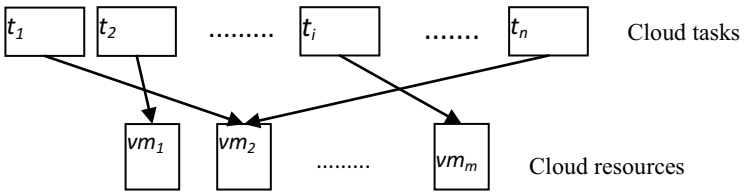

Figure 2 Task scheduling in cloud.

predefined set of constraints and measurements. In simple words,*n*number of tasks (T) is to be mapped (assigned) to m number of resources (R). The criteria like makespan, energy efficiency, tardiness, flow time, etc. are widely used to measure the performance of scheduling algorithms.

Similar to traditional parallel scheduling, cloud scheduling problem at the application layer is to focus on finding an "optimal" mapping $\mathbf{C{:}T \times VM \rightarrow F}$ which will assign n tasks $T = \{t_1, t_2, \ldots, t_n\}$ on to m available virtual machines $VM = \{vm_1, vm_2, \ldots, vm_m\}$ such that the fitness of objective functions $F = \{f_1, f_2, \ldots, f_z\}$ is either maximized or minimized collectively in weighting or individually with various constraints such as deadline, budget, etc. [4]. In case of scheduling problem at the virtualization layer, set T represents virtual machines to be scheduled and set VM represents physical machines. It is noted that cloud scheduling problem is an NP-hard. Fig. 2 illustrates the schematic view of task scheduling in the cloud where more than one task may have to share the same virtual machine. The cloud-specific scheduling objectives (objective functions)include makespan, cost, energy conservation, load balancing, resource utilization, security, fault tolerance, scalability, etc.

The scheduling problem usually transformed into some mathematical formulations e.g., linear programming (LP), integer linear programming (ILP), quadratic programming (QP), quadratic constraint programming (QCP), etc. Among these formulations, the most common is ILP problems [17] which are optimization problems where one has to find the best solution from many feasible solutions. The standard form of an ILP problem is as follows [17]:

$$\begin{array}{ll} & \min c^T x \\ \text{Subject to} & Ax \geq b \\ \text{and,} & x \geq 0 \end{array}$$

where x represents the vector of n unknown decision variables whose value is to be determined, $c(c_1, c_1, \ldots c_n)$ is a known vector of coefficients of respective variable, $b(b_1, b_2, \ldots b_m)$ is a vector of right-hand side(RHS) values of m inequalities, and $A(m \times nn$ is a two-dimensional known matrix of coefficients. The expression $c^T x$ which is to be minimized is called the objective function. Depending upon the variable being optimized, the optimization problems can be divided into two categories, namely, *continuous optimization* and *combinatorial optimization* problems [17] where former can have real values for variable x and the later can have only integer values.

Cloud scheduling problem also leads to an ILP problem which is proved to be an NP-hard optimization problem [18][19]. For instance, in [20] scheduling problem of mapping dependent task to VMs has been formulated for the solution (s) as follows:

$$\begin{array}{ll} minimize & C_{\max}(s) + \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} \\ subjectto & C_{\max}(s) \leq U(s) \\ & C(s) \leq B(s) \end{array}$$

where n and m are the numbers of tasks and machines respectively, $U(s)$, $B(s)$ and $C(s)$ are the number of overdue tasks, restriction on the budget for the tasks of schedule s, and total cost of s respectively. And, C_{ij} and $C_{max}(s)$ represents the cost of processing the ith task on the jth machine and the completion time of the last job i.e., makespan respectively.

3. TASK SCHEDULING ALGORITHMS AT APPLICATION LAYER: CHARACTERISTICS AND CLASSIFICATION

The resource management in the cloud at the application layer as well as the virtualization layer is handled through scheduling. The application layer scheduling involves two stakeholders, namely, *cloud service provider* and *consumer*. The cloud service provider provides the resources in the form of VMs on a rental basis to consumers who submit (schedule) their tasks over those VMs for processing. Both these stakeholders have their own set of QoS requirements. The consumer is concerned with the performance of various applications in terms of execution time, deadline, cost (budget), fault tolerance, security etc., whereas the service provider is more interested in efficient resource utilization and energy efficiency in the data center. Thus, the QoS requirements can be categorized into two groups, namely, *Consumer-desired* and *Provider-desired*. Efficient application scheduling helps to achieve a better quality of services for both stakeholders. The quality requirements of both categories have been shown in Figs. 3(a) and 3(b) as explained by Xhafa *et al.* [21] for clouds and Grid environments.

It is noted that some of these QoS parameters need to be either maximized or minimized. While transforming the scheduling problem into the linear program, choosing the QoS objective as a variable in a linear function or keeping it as a constraint depends upon the requirements, assumptions and the environment.

Apart from the various QoS requirements which are targeted or optimized by scheduling algorithms, there exist few other essential characteristics identified in the literature [22] that can be used to classify the algorithms in various categories which are as follows:

- Static/Dynamic

  The static scheduling is aware of the arrival time of tasks which is very hard to predict, however, gives less overhead at runtime. On the other hand, dynamic scheduling incurs

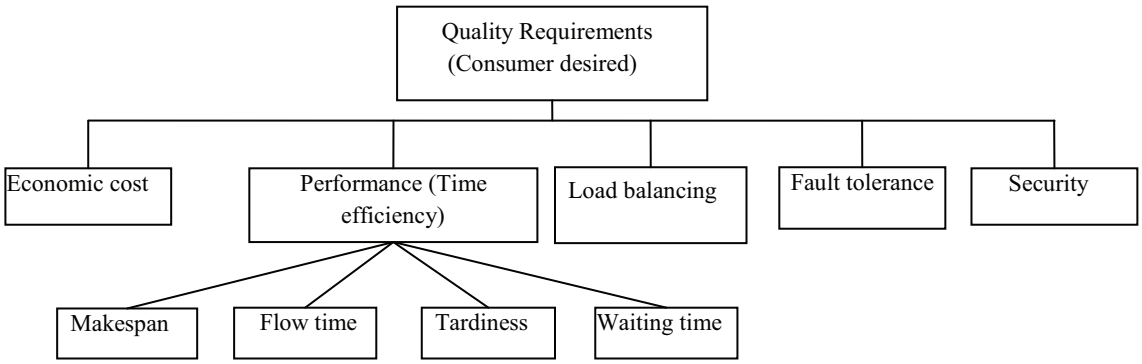

Figure 3 (a): Consumer-desired quality requirements.

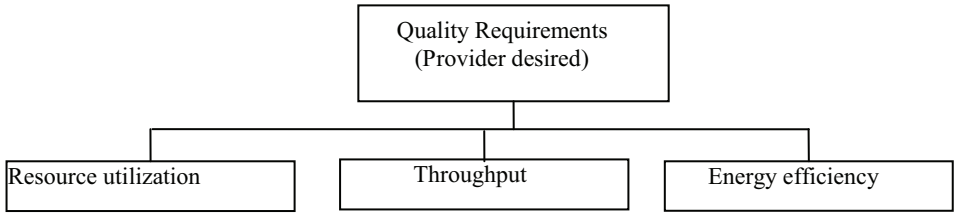

Figure 3 (b): Provider-desired quality requirements.

runtime overhead as the task arrival information is unknown at rum time. Nevertheless, dynamic scheduling better adapts to the timing changes [23][24]. In other texts, static and dynamic scheduling are also referred to as off line and online scheduling respectively. Min-min and Max-min algorithms are traditional examples of off line scheduling.

- Centralized/Distributed

  The centralized scheduling is controlled by a master processing unit for the collection of tasks and further sent to other workers for processing. On the other hand, in distributed scheduling, local schedulers manage the requests of resources and maintain the job states. Centralized scheduling is more efficient than distributed scheduling due to no overhead for maintaining the coordination among local schedulers, however, it suffers from single node failure.

- Preemptive/Non-preemptive

  The preemptive scheduling allows each task to be interrupted during the run-time and hence, resources can be released from the task. On the other hand, the resource can be released only on the completion of tasks in non-preemptive scheduling,

- Single/Multi-objective

  The single objective scheduling algorithm optimizes only one metric at a time whereas multiple metrics can be optimized in multi-objective scheduling [25]. The multi-objective optimization becomes more difficult when conflicting criteria are optimized.

3.1 Application Type

Job or application characteristics greatly influence the design principal of scheduling algorithms. The type of applications scheduled over VMs may vary from simple independent tasks to complex graph applications *e.g.*, social network analysis. The scheduling algorithms differ significantly on the basis of the application being scheduled due to later's essential intrinsic properties. For example, a MapReduce (MR) job is a set of several map and reduce tasks to be scheduled over the limited map/reduce slots on various VMs along with a constraint of completing all map tasks before reduce tasks start [26] This special characteristic drives the different methodology of MR scheduling algorithm over the cloud On the other hand, a scientific workflow application is a set of tasks, dependent on each other and needs to be executed according to that dependency [20]. While designing a scheduling algorithm that dependency should be taken into consideration. Such diverse characteristics of applications pose a great challenge while designing an efficient scheduling algorithm. We note that the different categories of applications that are usually scheduled over cloud include a set of independent tasks, simple or scientific workflow, MapReduce application, and most recently Big Graph application, etc

3.2 Fundamental Techniques Used in Designing Cloud Schedulers

Traditional techniques used in single or multiprocessor scenario takes exponential time in the cloud environment to find a feasible solution. Hence, other advanced techniques that find a sub-optimal or near-optimal solution in a given time frame are used to design cloud scheduling algorithms. Various Cloud schedulers available in the literature are essentially based on some fundamental technique. For example, some of them are based on particle swarm optimization (PSO) technique, whereas, some are based on problem-specific heuristic methods. Different fundamental approaches that are used in the context of cloud scheduling are as follows:

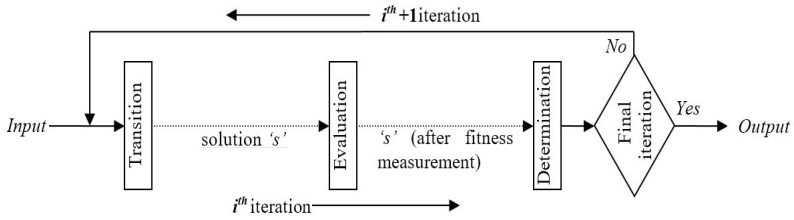

Figure 4 General steps in heuristic and meta-heuristic techniques.

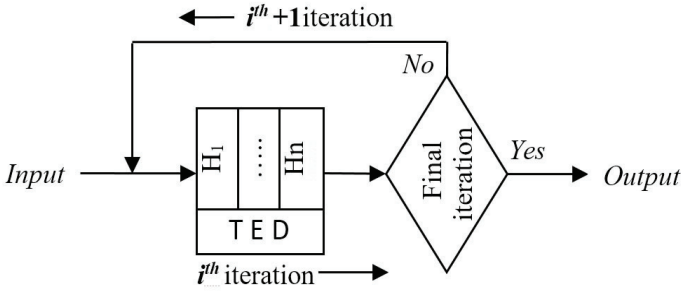

Figure 5 Steps hybrid heuristic techniques.

- **Traditional approach:** Several traditional scheduling techniques *e.g.*, FIFO, greedy approach, EDD [27], CPM[28], PERT [29], dynamic programming [30] and branch-and-bound [31] etc. may be used in a cloud environment. However, such techniques fail to work due to a large solution space and unable to find the optimal solution in polynomial time. These methods are generally much easier to implement in comparison to metaheuristic and hybrid scheduling algorithms as their design is based on one or few particular rules to manage and arrange the tasks.

- **Heuristic approach:** The heuristic techniques, on the other hand, are problem-dependent and can be adapted so as to consider the intricate and essential particularities of a given problem. However, being too greedy, the heuristic techniques usually get trapped in a local optimum and thus, fail to obtain a globally optimal solution. Because heuristics use "tactical guess" to find the possible solutions, there is a better chance to find an optimal result as compared to rule-based deterministic algorithms.

- **Meta-heuristic approach:** The meta-heuristics techniques are problem-independent and do not take advantage of any specificity of the problem. It is a special category of heuristic techniques. Here, the technique may adapt to temporary deterioration of the solution to explore better results. Though meta-heuristic is problem-independent, it is nonetheless necessary to fine-tune the intrinsic parameters of a problem in order to achieve an optimal solution.

- **Hybridheuristic approach:** The hybrid heuristic technique combines two or more heuristic or meta-heuristic algorithms into a single heuristic. Recently a new kind of hybrid heuristic technique has been developed known as hyper-heuristic which considers the space of heuristic or meta-heuristic techniques as a solution space. Indeed, the hyper-heuristic can be thought of as "heuristics to search for heuristics" or sometimes as "heuristics to generate heuristics."

The basic idea behind heuristic and meta-heuristic techniques is to use three key operators namely, *transition*, *evaluation*, and *determination* for searching large solution space on the convergence process. Fig. 4 shows the generic flow in heuristic and meta-heuristic techniques. Both these techniques operate in an iterative fashion. In each iteration, first, the transition operator creates the solution s by using the perturbative or constructive or both methods [43]. Thereafter, the fitness of solution s is measured by evaluation operator using a predefined measurement. Finally, the determination operator determines the next search directions based on the s from the transition and the evaluation operator.

The basic idea of the hybrid-heuristic algorithm is shown in Fig. 5, where two or more heuristics/meta-heuristics algorithms are combined to exploit the complementary advantages to find a better result. At each iteration, during the transition (T), evaluation (E), and determination (D) stage in the convergence process, any one of the participating heuristic/meta-heuristic H_i may be used.

In view of the above discussion, the scheduling algorithms at the application layer in cloud computing can be classified on the basis of (i) QoS objectives, (ii) Other essential characteristics e.g., static/dynamic, preemptive/non-preemptive etc., (iii) type of application scheduled, and (iv) fundamental technique used in designing. In this work, we classify all discussed scheduling algorithms on the basis of various fundamental techniques like heuristic, meta-heuristic and hybrid-heuristic used while designing the scheduling algorithms. A detailed taxonomy of discussed algorithms has been shown in Fig. 6.

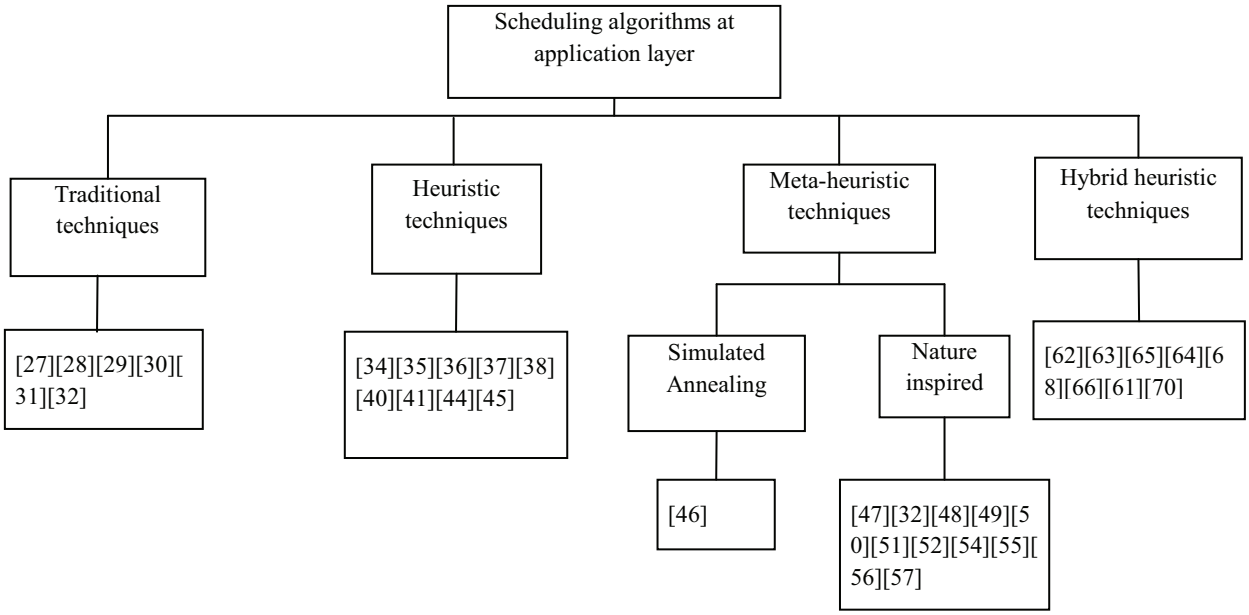

Figure 6 Taxonomy of various scheduling algorithms on the basis of fundamental technique used.

4. STUDY OF SCHEDULING ALGORITHMS AT THE APPLICATION LAYER

The scheduling at the application layer is important and must be efficient for both cloud consumer and service provider. An efficient scheduling technique helps in the optimal utilization of the available cloud resources. The problem cloud scheduling is complex and NP-hard due to which the traditional techniques like min-min [32], max-min [32], FIFO, round-robin, etc. may not achieve better performance. Therefore, such techniques may be either combined with other existing efficient techniques or improved as proposed by Kumar *et al.* [32] and Gang Liu *et al.* [33]. In this section, we present the main study of various cloud scheduling algorithms classified according to taxonomy given in Fig. 6. We devote a separate section for each fundamental technique. At the end of each section, we provide two tables which summarize the various attributes of the discussed algorithms.

4.1 The Heuristic Approach

The heuristic approach is problem-dependent unlike meta-heuristic techniques and can be adapted in order to consider the intricate and essential particularities of a given problem. Singh *et al.* [34] proposed a score-based fault-tolerant workflow scheduling heuristic with a deadline as a constraint. The proposed scheme reduces the failure rate and execution time with a cost that is manageable by the user. The capabilities of resources have been represented by score values that are used for the allocation of a workflow task to resources. In the scheme, the list of the workflow tasks (DAGs) is initially submitted to get the available virtual resources from the data center. Afterward, the entire workflow is imposed by a user deadline. Thereafter, the final score of VM's is obtained from the components of minimum sub-scores and then a VM with the lowest sub-score is picked that satisfies the threshold of the task. Lastly, the task is assigned to the selected VM to execute the given task within a deadline. In case, the VM fails to meet the deadline, the selection of a new VM with the next minimum score is carried out. The process continues until all the tasks are mapped to the VMs.

Bitten court and Madeira [35] proposed a scheduling algorithm HCOC (Hybrid Cloud Optimized Cost) for a hybrid cloud environment to minimize the cost and makespan of a workflow application. The algorithm decides about the resources that should be leased from the public cloud and aggregated to the private cloud. The motive here is to provide enough processing power in order to execute a workflow within a given time period. Firstly, the selected tasks are rescheduled and thereafter the resources from a public cloud are taken along with consideration of both price and performance of the resource. The multi-core awareness facility and known cost of a resource can minimize the overall makespan as desired by the user. Moreover, the adaptability of the HCOC algorithm to handle cost instead of a deadline makes it more flexible.

Verma and Kaushal [36] proposed a Deadline and Budget Distribution based Cost-Time Optimization (DBD-CTO) algorithm which minimizes the cost and execution time while meeting the user-defined deadline and budget constraints. They divide the workflow tasks into two groups of tasks. The tasks with more than one parent or child task are categorized as *synchronization tasks* and rest are categorized as *simple tasks*. The partitioning of workflow is done in a way so that a group of simple tasks is executed sequentially between two synchronization tasks. The workflow actually starts executing when the calculated values of execution time and cost are less than the user-defined deadline and budget. Thereafter, the overall deadline and budget are distributed into every partition of tasks, proportional to their minimum processing time and cost. Finally, all service lists are arranged in decreasing order of their cost. A service is chosen to execute tasks such that the processing cost and execution time are lesser than the partition's deadline and budget value.

Poola *et al.* [37] used spot and on-demand instance pricing models while scheduling workflow tasks on VMs to reduce the overall execution cost, thereby, meeting the user-defined

deadline. The main advantage of the approach is to tolerate early termination of spot instances and variations in the performance of cloud resources. In the scheme, for every ready task, the critical path is evaluated and slack time is computed, i.e., the time difference between critical path time and deadline. Whenever the slack time decreases due to performance variations or failures in the system, a checkpointing and bidding strategy is applied to reduce the cost and hence, meet the deadline.

Bessai *et al.* [38] proposed three workflow scheduling schemes to optimize the execution time and total cost of utilizing the resources. The first scheme uses several allocation techniques to minimize the communication and execution cost while calculating the completion time of each solution. On the other hand, the second scheme aims to minimize communication and execution time while calculating the computation cost of each solution. The third scheme is based on the pare to solution obtained by the first two solutions and is called the cost-time based approach. Thus, in the third approach, only non-dominated solutions are selected by using the cost and time-based approaches.

Among all the research work that has been conducted to achieve fault tolerance in distributed systems like Grid and Cluster, scheduling plays a significant role [39]. However, very few fault-tolerant scheduling schemes have been studied in cloud computing taking virtualization and elasticity into consideration. To address these issues, Wang *et al.* [40] extended the primary-backup model and presented a fault-tolerant mechanism FESTAL (Fault-tolerant Elastic Scheduling algorithms for real-time TAsks in cLouds) to incorporate the cloud-specific features. The elastic resource provisioning method is used for real-time tasks to achieve both, high resource utilization and fault tolerance in the cloud.TheFESTAL comprehensively addresses the issue of reliability, elasticity, and schedulability of virtualized clouds.

The cloud infrastructure is also used for big data processing using Hadoop MapReduce framework. Its performance is heavily governed by its scheduler, which implicitly assumes that tasks make progress linearly and the cluster is homogeneous i.e., all cluster nodes are similar in terms of computational power, disk, and network I/O bandwidth. However, the performance of Hadoop's scheduler degrades severely in a heterogeneous environment. The most common place where this heterogeneity occurs is virtualized cloud computing infrastructure like Amazon EC2, Google, etc.

In the context of Hadoop MapReduce, slow tasks (map or reduce) due to faulty hardware or heavy load on that particular node are called stragglers. To address the straggler detection problem while scheduling the MapReduce application over the cloud, Zaharia *et al.* [41] proposed a scheduling algorithm called as Longest Approximate Time to End (LATE) that is highly robust to heterogeneous computing system like a cloud. LATE decides when to speculatively re-execute tasks that appear to be stragglers to improve response time in a cloud environment. It prioritizes speculative tasks (unlike native Hadoop scheduler which considers all stragglers equally low), selects fast nodes for execution, and binds the number of speculative tasks using a threshold to prevent thrashing.

A simple heuristic has been proposed in LATE that work well in practice. The algorithm first calculates the *ProgressScore* (PS) of a task as shown in Eq. 1 (like Hadoop native scheduler) where M is the number of key/value pairs that have been processed and N is the number of key/value pairs that need to be processed in total for map task. In the same manner, M' is the number of key/value pairs that have been processed and N' is the number of key/value pairs that have been processed in any particular phase of reduce task. Further, it calculates the *ProgressRate* of each task as *ProgressScore/T*, and then the new heuristic, "time to completion" or Approximate Time to End (ATE) of the task is estimated as *(1-ProgressScore)/ProgressRate*. The heuristic serves to prioritize the stragglers *i.e.*, tasks with high "time to completion" values are speculatively re-executed first. A different technique to estimate the completion time may also be incorporated into LATE. Furthermore, LATE also improves the performance of speculative execution in a homogeneous environment.

$$PS = \begin{cases} M/N & for\ map\ task \\ 1/3^{*}\left(K + M'/N'\right) & for\ reduce\ task \end{cases} \quad (1)$$

To improve the LATE algorithm, Chen *et al.* [42] and Xiaoyu *et al.* [43] proposed Self-adaptive MapReduce Scheduling Algorithm (SAMR) and Enhanced Self-Adaptive MapReduce scheduling (ESAMR) respectively, with a better heuristic to identify the stragglers. Yang *et al.* [44] further improved the original speculative execution in Hadoop (called Hadoop Speculative) and LATE scheduler in a heterogeneous cloud environment by proposing a new scheduling scheme known as Adaptive Task Allocation Scheduler (ATAS). It employs an efficient and more accurate heuristic called *TimeToEnd* for each task in order to trace stragglers. The motive of this improved method is to increase the success ratio of backup tasks that consequently increases the system's ability to respond in an efficient manner. The nodes are divided into *QuickNode* and *SlowNode*. The *QuickNode*is always given priority while the allocation of backup tasks. The authors performed three simulation experiments and concluded that ATAS effectively enhances the performance of Hadoop framework in a heterogeneous cloud computing environment.

Security in the cloud computing environment is one of the most important issues as sensitive data may get leaked to unauthorized persons. Secure scheduling prevents the allocation of tasks and associated data to vulnerable machines. Abazari et al. [45] proposed a heuristic algorithm for tasks scheduling which is based on the task's security requirements and completion time. To quantify tasks security requirements, the authors introduced task security sensitivity measurement. Besides this, they also proposed a new attack response to tackle some security threats.

Table 1 summarizes the various QoS metrics used during the evaluation of individual algorithm discussed in this section. In addition, Table 2 analyzes the discussed algorithms on the basis of objectives, SLA adherence, strengths/weakness, experimental/environments scale, and target application type, etc.

4.2 The Meta-heuristic Approach

The meta-heuristic technique has the following two categories:

(i) *simple e.g.*, simulated annealing

Table 1 Analysis of metrics measured during the evaluation of heuristic algorithms.

Algorithm	Quality metrics measured	Comparison done with
Score based [34]	Execution time, Cost, Failure rate	Other non-score based algorithms
HCOC [35]	Makespan, Cost	Greedy approach
DBD-CTO [36]	Execution time, Cost	None
Poola *et al.* [37]	Cost, Failure rate	Baseline Algorithms [37]
Bessai *et al.* [38]	Time, Cost	None
FESTAL [40]	Deadline, Resource utilization, Fault tolerance	Elastic First Fit and two variants of FESTAL
LATE [41]	Running time,	Hadoop native
ATAS [44]	Execution time, Average throughput, Latency	LATE, Hadoop speculative
MOWS [45]	Makespan, security risk, security threat	HEFT, SAHEFT

(ii) *nature-inspired e.g.*, particle swarm optimization (PSO), ant colony optimization (ACO), genetic algorithm (GA), etc.

The simulated annealing (SA) is a probabilistic procedure to approximate the global optimum of a given objective function. Precisely, it is a meta-heuristic to estimate global minimum or maximum in a given large search space. The basic idea of SA has been derived from the physical annealing process in metallurgy. The technique involves heating and controlled cooling of a material to increase the size of its crystals and reduce the defects. The so-called physical annealing has three stages, (i) *Heating* to enhance the thermal motion of particles, (ii) *Isothermal* to exchange heat with the surrounding environment and (iii) *Cooling* to make the thermal motion of the particle weaken and become more orderly. Inspired by SA, Xi Liu *et al.* [46]proposed a task scheduling mechanism to overcome the shortcomings of the local optimum search method. The method uses a greedy approach to generate an initial value. In the heating stage, the temperature is raised sufficiently and afterward, a given set of rules is used to generate a new value. In case, the new value is either better than the original value or possess an acceptable probability;the new value is replaced by original value until cooling stage. Compared with traditional algorithms, the task scheduling based on SA meets the user's requirement as well as enhances the overall performance of the system.

Analysis of metrics measured during the evaluation of heuristic algorithms.

The application of genetic algorithm (GA)as cloud scheduling technique can be traced back to Early 2009 when Zhao *et al.* [47] proposed a scheduler to map independent and divisible tasks to cloud resources with makespan as the objective. In the scheme, M tasks are scheduled over N cloud resources with a simple chromosome encoding as already depicted in Fig. 2 (section 2), assuming the number of tasks as its length. Each gene is represented as an integer i where $i \in \{1, 2, 3, \ldots, M\}$. Further, each index of the resource is represented as j *where* j $(1, 2, \ldots, N)$, indicating that i^{th} task T_i has been scheduled on the j^{th} resource R_j. Therefore, the use of GA is simple and better to formulate cloud scheduling problems as proposed by Kumar *et al.* [32] and Junwei *et al.* [48]**.**The work proposed by Kumar *et al.* [32] combines GA with Max-Min and Min-Min to improve the speed of the algorithm and population initialization, thereby, minimizing the makespan.

In classical GA method, the initial population (essentially, schedule in this context) is generated randomly and may not always be fit. Also, whenever mutated, there is a minimal chance that the initial populations produce a better schedule. Hence, using min-min and max-min with GA while generating an initial population, a fit schedule can be produced resulting in better schedules whenever mutated. Junwei *et al.* [48] used the same encoding scheme as Zhao *et al.* [47] and proposed a modified genetic algorithm (MGA) to schedule the cloud resources. The parameters such as average makespan, total makespan, user cost have been considered and prove MGA to be more efficient for cloud computing. Nevertheless, the scheme of chromosome encoding is typical. However, the same is widely used for cloud resource scheduling.

Ant colony optimization (ACO) is another popular nature-inspired meta-heuristic technique used in cloud scheduling. In Fig. 7, a general framework is illustrated where ACO is used to schedule user tasks on cloud resources [4]. In its simplest form, M steps are used by each ant in order to construct a solution. The heuristic information and pheromones are used by an ant to select the suitable resource R_j in the i^{th} step for scheduling the i^{th} task T_i. Further, after executing M steps, all tasks are scheduled with different resources.

Based on the same scheme aforementioned, Banerjee *et al.* [49] and Liu *et al.* [50] schedule M tasks one by one to the resources of the cloud as only a single task can be scheduled on any resource at each step. Banerjee *et al.* [49] modify the pheromone update scheme according to different time slots of cloud service. In the scheme, each antis positioned on a starting node and a state transition rule is applied to build an iterative solution. In addition, a local pheromone has been used to update rule until all ants built a complete solution. The analysis, however, does not consider fault-tolerance issues. Liu *et al.* [50] use the heuristic information based on the user's QoS criteria like cost, response time, system reliability and security to guide ant to select an optimal resource. The scheduling algorithm is designed to schedule service flow with several QoS requirements as mentioned above. The end users are permitted to define QoS threshold in software level agreement (SLA). In order to ensure the QoS, the default rate is used to denote the ratio that may be dishonored by a cloud service provider. An SLA monitoring module has been introduced to keep a check on the running state of cloud services.

In literature, the ACO-based approach was also used by Zhu *et al.* [51] to optimize user cost, network bandwidth, makespan, and system reliability while scheduling applications over cloud resources. According to different QoS metrics, the tasks are

Table 2 Analysis of various heuristic scheduling algorithms.

Algorithm	Objectives	SLA constraints	Strengths/Weakness	Experimental environment/Scale	Application type	Type of cloud	Mathematical formulation[a]
Score based [34]	Cost	Deadline	• capabilities of hardware resources are represented by score values	CloudSim/5–25 tasks	Workflow	Public cloud	No
HCOC [35]	Cost, makespan	Deadline	• Considers the hybrid cloud environment • Improves the budget effectiveness within a deadline	Testbed environment/NA	Workflow	Hybrid	No
DBD-CTO [36]	Cost, makespan	Deadline and budget	• Does not optimizes any QoS parameter rather achieves two constraints: deadline and budget.	Simple simulation in JAVA/NA	Workflow	Public cloud	No
Poola *et al.* [37]	Cost, Makespan & Fault tolerance	Deadline	• Uses two different pricing models, spot and on-demand instances	CloudSim/1K tasks of LIGO workflow, 9 VMs	Scientific workflow	Public cloud	No
Bessai *et al.* [38]	Cost Makespan	None	• Optimizes two conflicting criteria, cost and time simultaneously	Simulator not specified/50–1000 tasks, 5–100 VMs	Workflow	Public cloud	Yes
FESTAL [40]	Fault tolerance, resource utilization	Deadline	• Exploits the traditional primary-backup model for fault-tolerance	CloudSim/5K–40K tasks	Independent tasks	General	No

[a]Whether the cloud scheduling problem has been formulated as integer linear program (ILP) or other mathematical model

Table 2 Continued

LATE [41]	Response time	None	• Improves the response time through speculative execution • employs a static procedure to calculate the progress of tasks to identify stragglers	Virtualized cluster over Amazon EC2 and local physical cluster/15–871 VMs, WordCount job	MapReduce	General	No
ATAS [44]	Response time	None	• Improve the LATE algorithm by better detection of stragglers • Back-up tasks are scheduled only on *QuickNodes*	Virtualized cluster over private physical machines/10–100 VMs, 3 MR jobs	MapReduce	General	No
MOWS [45]	Computation time, transmission time, security risk	None	• Utilizes the concept of list scheduling to propose its own heuristic method	Workflow Sim 20 PMs, 100 VMs 30–1000 tasks	Scientific workflow	General	No

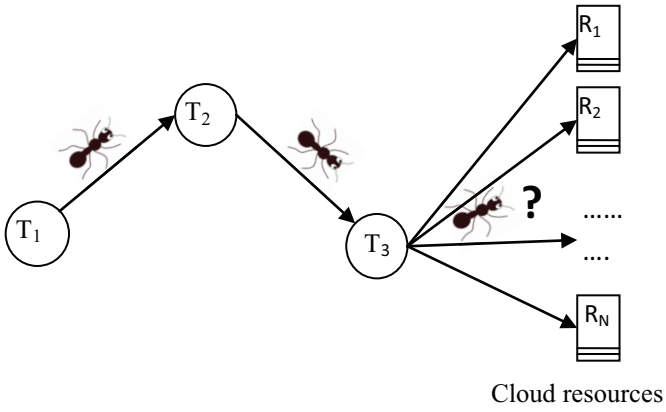

Figure 7 Illustration of ACO techniques for cloud resource scheduling [4].

first classified in different categories and afterward bounded via the ACO optimization to the cloud resources in each category. Nishant *et al.* [52] use an ACO-based algorithm to schedule the cloud resource in order to optimize the load balance of different nodes. The modified algorithm is better in comparison to an original approach where each ant builds their individual result set. Here, the ants continuously update a single global result set rather than updating their own local result set. The main advantage of the method lies in its detection of overloaded and under-loaded nodes and so carrying out operations based on the identified nodes. Moreover, the modified ACO based approach can choose a node with a maximum number of neighboring nodes. This facilitates the ant to find more nodes that are overloaded or under loaded while traveling in the optimal direction. Thus, some of the load from heavily loaded VMs may be redistributed to the light-loaded VMs.

In addition to ACO and GA, particle swarm optimization (PSO)is also a meta-heuristic technique as it makes few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions [53]. PSO is an effective tool for scheduling cloud task on cloud resources and offers faster convergence than other meta-heuristic algorithms. A simple variation of the PSO procedure works by having a population (called a swarm)of candidate solutions (called particles). These particles are moved around in the search-space using few simple formulae. The movements of the particles are directed by their own best-known location in the search-space as well as the entire swarm's best-known location. Whenever an improved position is being discovered, the particles participate and guide the movement of the swarm. The process is repeated that eventually lead to a satisfactory solution.

Pandey *et al.* [54] use the PSO technique to schedule cloud tasks employing the same GA encoding scheme as described in [47]. In the method, the task number is assumed to position length of particles and the integer value indicates the cloud resource is executing the corresponding task. The approach considers, both, data transmission and computation cost. The total cost of execution is obtained by varying the cost of computing resources and communication cost between resources. The proposed heuristic is generic in nature due to the fact that any number of resources and task can use it only by increasing the particle dimension and number of resources. Later, Rodriguez and Buyya [55] proposed a PSO based workflow scheduling algorithm in a public cloud to minimize the overall execution cost while meeting a user-defined deadline. Besides a scheduling algorithm at the application layer, the proposed scheme also discusses a resource provisioning algorithm at the virtualization layer. The scheduling algorithm attempts to minimize the Total Execution Cost (*TEC*)of the schedule. The formulated problem has been shown in Eq. 2 where LET_i and LST_i are *Lease End Time* and *Lease Start Time* respectively of the i^{th} virtual machines with τ as a unit of time. Furthermore, *TET* is total execution time and δ_w is a deadline associated with workflow (W).

$$\begin{array}{ll} minimize & TEC = \sum_{i=1}^{n} \cos t_i * floor\,(LET_i - LST_i/\tau) \\ subjectto & TET \leq \delta_w \end{array} \tag{2}$$

In this scheme, the real number is rounded to integer number to indicate the resource index on which the workflow is scheduled. It is worth mentioning that the features of the resources are not reflected by the index of the resource. Therefore, the particles may fly randomly, in case, learning has been made from the resources index.

Dhinesh *et al.* [56] designed an algorithm named as Honey Bee Behavior inspired Load Balancing scheme (HBB-LB). The algorithm aims to maximize the throughput of the system and achieves a balanced load across virtual machines. In addition, along with balancing the load of VMs, the priorities of tasks are considered. The tasks are removed from heavily loaded VMs so that the overall waiting time of the task in the queue can be minimized. The removed tasks from these VMs are treated as honey bees that act as information updater globally.

Wen et al. [57] proposed a secure Multi-Objective Privacy-Aware workflow scheduling algorithm (MOPA) which minimizes both execution time and cost with a data privacy protection constraint. Given a set of workflow instances in the cloud environments the authors have modeled a multi-objective optimization problem as shown in Eq. 3.

$$Minimize \quad F = (Twtotal, CwtotalsatisfyingP \tag{3}$$

Where $Twtotal$ and $Cwtotal$ are total execution time and monetary cost respectively, and P is the set of privacy protection constraints. MOPA proposes a problem-specific encoding strategy which is based on the currently-generated workflow schedules, candidate lists of VM instances, and related privacy protection constraints. The authors compare the proposed scheme with NSGA-II and MOPSO which were modified with

Table 3 Analysis of metrics measured during the evaluation of heuristic algorithms.

Algorithm	Quality metrics measured	Comparison done with
Zhao *et al.* [47]	Resource utilization, Time utilization	NA
IGA [32]	Makespan	Standard Genetic Algorithm
MGA [48]	Completion time, Cost	AGA, CGA
Modified ACO [49]	Makespan	Classic ACO
Liu *et al.* [50]	Response time, Cost, Fault tolerance, Security	None
Zhu *et al.* [51]	Makespan	Random Distribution Algorithm
Pandey *et al.* [54]	Cost	Best Resource Selection (BRS)
Rodriguez and Buyya [55]	Makespan, Cost	IC-PCP [58], SCS [59]
HBB-LB [56]	Makespan, Load balancing	Weighted Round Robin, FIFO, Dynamic Load Balancing (DLB) [60]
Nishant *et al.* [52]	Load balancing	Classical ACO
MOPA [57]	Average hyper volume, Average coverage	NSGA-II, MOPSO

their encoding strategy. All discussed algorithms in this section have been summarized in Table 3 and 4.

4.3 The Hybrid-Heuristic Approach

Every heuristic and meta-heuristic scheduling algorithm has a tradeoff in terms of performance. For example, ACO technique may provide a better schedule than other traditional scheduling algorithm in terms of the total cost. However, it takes more computation time. Hence, in order to overcome the limitations of an individual technique, the latest approach in the field of cloud scheduling is to combine the two or more heuristics or meta-heuristic techniques into a single heuristic method. This kind of integration may compensate for the intrinsic weak points of the specific heuristic algorithm. For instance, Wen *et al.* [61] improve the resource utilization ratio by combining PSO and ACO schemes. The proposed scheme takes a longer computation time at each iteration of the convergence process, however, it has a higher chance to find a better result than a single heuristic.

Delavar and Aryan [62] propose a hybrid scheduling algorithm GMSW to map workflow tasks (DAG) on the cloud resources having various communication cost. It considers the suitable distribution of the workload on VMs and helps in reducing the number of GA operations via making an optimized initial population. The algorithm uses two evaluation functions in order to obtain a solution. The first function measures the priority of every task in the workflow DAG and the second function evaluates the value of the generated solution. Due to the hybrid nature of the algorithm, the optimal solution can be achieved early. Further, the searching process is done on the basis of the failure frequency and workload, led by special mutation method, considering the most effective task and the resources reassignment.

In recent years, multi-objective cloud scheduling has emerged as one of the major challenges. The concept is based on the optimization of more than one QoS parameters simultaneously. Though the problem is NP-hard, evolutionary computing techniques $i.e.$,meta-heuristics proved to be efficient with a minimum time overhead. However, the multi-objective scheduling becomes difficult whenever one parameter has to be minimized and other parameters need to be maximized. In literature, a multi-objective scheduling algorithm has been proposed by Yassa *et al.* [63] and Mezmaz *et al.* [64] where energy is one of the scheduling objectives. In both algorithms, Dynamic Voltage and Frequency Scaling (DVFS) technique have been used to minimize overall energy consumption. Yassa *et al.* [63] used a hybrid PSO scheme to optimize makespan, cost as well as energy simultaneously. Also, the heterogeneity of cluster nodes has been taken into consideration. Using the DVFS scheme, the processor can be operated on the different supply voltage. However, the clock cycles are sacrificed in terms of speed. Therefore, energy can be saved only by decreasing the execution time that may compromise the quality of schedules and energy. The hybrid scheme $i.e.$, DVFS Multi-Objective Discrete Particle Swarm Optimization (DVFS-MODPSO) produces a set of non-dominated solutions in order to evaluate the user preference. Accordingly, a schedule can be selected to achieve the QoS requirement. Mezmaz *et al.* [64] presented a new parallel multi-objective (bi-objective) hybrid genetic algorithm to schedule the precedence constraint application like DAGs over heterogeneous computing system $e.g.$, cloud computing. Being bi-objective in nature, the algorithm optimizes makespan (completion time of the last task in DAG) as well as energy using DVFS technique. In the experimental study, the results exhibit a reduction in energy consumption 47.5% and makespan by 12%.

Another important QoS parameter in hybrid heuristic is load balancing A better workload balanced system always enhances the overall throughput and energy efficiency. Delavar and Aryan [65] proposed a scheme to optimize load balancing, speed-up ratio and makespan. The algorithm combines GA with Round Robin and Best Fit techniques. Hence, the iteration of GA operations is decreased while executing the algorithm

Table 4 Analysis of various meta-heuristic scheduling algorithms.

Algorithm	Objectives	SLA constraints	Strengths/Weaknesses	Experimental/Environment Scale	Application type[a]	Technique	Type of cloud	Mathematical formulation
Zhao *et al.* [47]	Makespan	None	• The algorithmic logic is simple.	NumericalSimulation/ 2 tasks, 2 resources	Independent tasks	GA	Not specified	Yes
IGA [32]	Makespan Not specified	None	• The computing speed gets accelerated by combining two greedy strategies Min-Min and Max-Min with standard GA	CloudSim/10–40 tasks 10–40resources	Independent tasks	Improved GA	Not specified	No
MGA [48]	Makespan, Cost	Cost	• Uses an adaptive strategy for GA parameters by combining makespan and cost together	CloudSim/50 tasks, 50 resources	Independent tasks	Genetic algorithms	Public commercial cloud	No
Modified ACO [49]	Makespan	None	• Optimizes makespan by adding time slot information into pheromone • Fault tolerance issues have not been considered	Google AppEngine, MS Live Mesh/25 kinds of tasks	Independent tasks	ACO	Not specified	Yes
Liu *et al.* [50]	Reliability, response time, cost, and security	None	• Optimizes multiple QoS by combining different user objectives into one by weights	NumericalSimulation/ 10 tasks 10–50 resources	Service flow	ACO	Public commercial cloud	Yes
Zhu *et al.* [51]	Makespan	None	• Considers priority of the tasks while scheduling to improve response time	CloudSim/20–100 tasks, 8 resources	Independent tasks	ACO	Not specified	No

[a]Type of application for which scheduling algorithms has been designed

Table 4 Continued

Pandey *et al.* [54]	Cost	None	• PSO encoding does not take the differences in resources into consideration.	Amazon Cloud/5 tasks, 3 resources	Workflow	PSO	Public commercial cloud	Yes
Rodriguez and Buyya [55]	Cost	Deadline	• Considers both, the computational and communication cost • Features of the resources are not reflected by the index	CloudSim/50–1,000 tasks, 6 resources	Scientific workflow	PSO	Public commercial cloud	Yes
HBB-LB [56]	Load balancing	None	• Exploits the behavior of honey bee • Does not consider dependent tasks *i.e.*, DAGs	CloudSim/10–40 tasks 3–7 VMs	Independent tasks	Honey Bee Behavior (HBB)	General	Yes
Nishant *et al.* [52]	Load balancing	None	• Heavy-load node is detected by ants and some load is shifted to light-load node • No experimental evaluation has been done	Theoretical proof only	Independent tasks	ACO	Not specified	No
MOPA [57]	execution time and cost	Privacy Protection	• Proposes a secure multi-objective scheme to prevent the data leakage. • Exploits the GA to devise the problem-specific encoding and population initialize	CloudSim 12 VMs 10–100 workflows	Workflow	GA	Hybrid	Yes

Table 5 Analysis of metrics measured during the evaluation of heuristic algorithms.

Algorithm	Quality metrics measured	Comparison done with
GMSW [62]	Makespan, Fault tolerance/Failure rate/Reliability, Speedup	GVNS [72] CMMS [73] and LAGA [74]
HSGA [65]	Makespan, Load balancing, Speedup	LAGA [74] NGA [75]
DVFS-MODPSO [63]	Makespan, Cost, Energy efficiency	HEFT
Mezmaz et al. [64]	Makespan, Energy efficiency, Speedup	ECS [76]
HHSA [68]	Makespan, Waiting time	FIFO, fair scheduler
FUGE [66]	Makespan, Cost, Load balancing	Classical ACO, MACO [77]
ACO-PSO [61]	Time, Resource utilization	Classical ACO
Hybrid [70]	Speed up, Energy efficiency	Classical ACO

with an optimized initial population. Firstly, the impact of tasks on other tasks is analyzed and prioritized accordingly in the complex graph. Afterward, Best-Fit and Round Robin are combined to generate an optimal initial population. M Shojafar *et al.* [66] combined the FUzzy theory with the GEnetic algorithm and proposed a hybrid scheme called FUGE to balance the workload among VMs while optimizing execution time and cost. In the scheme, Standard Genetic Algorithm (SGA) is modified with a combination of fuzzy theory in order to develop a fuzzy-based steady-state genetic algorithm which may improve the performance in terms of makespan. The jobs are assigned to the available resources by considering the virtual machine's memory, bandwidth, processing speed, and job lengths. Authors formulated the job scheduling problem into a Linear Programming (LP) model as shown in Eq. 4, where the objective function is to minimize the total time required to finish all jobs subjected to eight constraints [66].

$$\min\left(f_i, R_j\right)\sum_{i=1}^{n}\sum_{j=1}^{m} d_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{m} L_{tot}(j) \Big/ A_{ij} f_j + {}^{L_i} \Big/ B_{ij} R_j \qquad (4)$$

Recently, hyper-heuristic techniques are gaining popularity indifferent research domains. In these schemes, two or more heuristic algorithms are combined [67] with two additional operators namely, low-level heuristic selection (LLH) and acceptance operator. LLH selection operator determines the selection of a heuristic algorithm whereas acceptance operator determines the timing to select a new heuristic algorithm. The hyper-heuristic also attempts to use two or more heuristics during the convergence process like other hybrid-heuristic schemes. However, it uses "one and only one" heuristic algorithm at each iteration. Hence, it is fundamentally different from the so-called hybrid heuristic algorithm which uses more than one heuristics (low-level) at each iteration, thus, requiring a much longer computation time. In the existing literature, Tsai *et al.* [68] proposed a hyper-heuristic scheduling algorithm (HHSA) to find better scheduling of task on cloud computing systems. The technique automatically determines the appropriate low heuristic algorithm (SA, GA, ACO, PSO) to be used with the help of two detection operators, namely, *diversity* and *improvement*. These low-level heuristic algorithms are used in finding a better candidate solution. Further, the solution is optimized through low-level heuristic by using perturbation operator to improve the performance in terms of makespan. The proposed "hyper-heuristic" technique exploits the strengths of every low-level algorithm keeping the computation time lower by executing one low-level algorithm at a time. The proposed HHSA algorithm has been validated through CloudSim and real Hadoop cluster.

Wen *et al.* [61] attempted to improve the resource utilization ratio desired by a cloud service provider and proposed a hybrid scheme consisting of ACO and PSO. The ACO is used as the main procedure to select appropriate resources for various tasks. The pheromone is associated with the resource node. Whenever a resource node is selected for a newly arrived task, the pheromone on the current resource node is reduced. The ACO as an individual technique achieves local optima and may terminate prematurely. Therefore, the PSO process is hybridized with ACO in order to maximize resource utilization. However, to calculate position and particle velocity, crossover and mutation operations are used to combine the search information of the individual best solution, global best solution and the particle itself.

Cuckoo Search (CS) is another meta-heuristic optimization technique developed by Yang and Deb [69] in 2009 which was motivated by the obligate brood parasitic activities of some cuckoo species in combination with the Levy flight actions of some birds and fruit flies. Apart from the population size n, The Cuckoo search uses a single parameter and hence it is very easy to apply in a wide variety of scenarios. Babukarthik et al. [70] and Navimipour et al [71] employed the CS algorithm in combination with other meta-heuristics ina various cloud scheduling application. Particularly, Babukarthik et al. [70] combined the advantages of ACO with the CS technique and proposed a hybrid algorithm to schedule the tasks to save energy. In the scheme, ACO is used as the main framework where CS is used to find the next resource for the task instead of heuristic information. Table 5 and 6 present the summary of discussed algorithms in this section.

5. OPEN CHALLENGES AND RESEARCH DIRECTIONS

The existing schedulers available in the literature for the application layer of cloud capture much of the aspects. Sometimes they

Table 6 Analysis of various hybrid-heuristic scheduling algorithms.

Algorithm	Objectives	SLA Constraints	Strengths/Weakness	Experimental environment and scale	Application type	Technique	Type of cloud	Mathematical formulation
GMSW [62]	Makespan	None	• Best-Fit, Round-Robin and a bi-directional tasks prioritization methods are merged to make a good initial population	Not mentioned	Workflow	Hybrid using GA	General	Yes
HSGA [65]	Makespan, load balancing	None	• Virtual list of resources and their updated properties are used to make a goal-oriented initial population.	Not mentioned	Workflow	Hybrid using GA	General	Yes
DVFS-MODPSO [63]	Makespan, cost & energy	None	• Proposes a multi-objective workflow scheduling which minimizes energy consumption using DVFS	The simulation tool is not mentioned	Workflow	Hybrid PSO	General	Yes
Mezmaz *et al.* [64]	Makespan & energy	None	• The bi-objective algorithm exploits the dynamic voltage scaling (DVS) technique	Grid of 3 clusters	Precedence constraint (DAGs)	Hybrid of GA and ECS	General	Yes
HHSA [68]	Makespan	None	• Combines multiple heuristics and develop a new methodology called hyper-heuristic to minimize makespan	CloudSim, single-node Hadoop and multi-node virtualized Hadoop	Scientific workflow	Hyper-heuristic	General	No

Table 6 Continued

FUGE [66]	Makespan, load balancing, cost	None	• Include fuzzy theory in a scheduling problem	CloudSim	Independent tasks	Hybrid (fuzzy theory + SGA)	General	Yes
ACO-PSO [61]	Utilization	None	• Combines ACO with PSO in which ACO selects resources, and PSO avoids prematurity	Numerical Simulation using MATLAB 7.0 50–400 tasks	Independent tasks	ACO+PSO	General	No other than optimization formulation
Hybrid [70]	Energy consumption	None	• Combines ACO with Cuckoo search • ACO is used as framework and Cuckoo Search(CS) as heuristics, but ignore the task's feature	Xen 1–256 tasks 1–10 resources	Workflow	ACO+ Cuckoo Search	General	No other than optimization formulation

come up with some heuristics to overcome the problem while some other times they exploit the complementary advantages of heuristic and meta-heuristic techniques to find a better result. However, there is still some scope for improvement. In this section, we will explore some aspects that must be examined to improve performance. Followings are some research directions.

Real-Time Scheduling

Scheduling of real-time workflow applications has not been taken much attention in a cloud environment. Traditional real-time scheduling on multiprocessor, Grid and cluster environment have been studied extensively in the available literature but achieving strict time requirement in virtualized and scalable cloud environment imposes great challenge. These time-dependent and mission-critical application cannot bear deadline miss. When and where to schedule real-time application efficiently so that it may acquire its deadline is a complex task to be still addressed.

Dynamic Scheduling

As the cloud resources (whether physical infrastructure or number and type of VMs) and the requests of the user can change dynamically, scheduling scheme must be smart enough to adapt to a changing environment in real-time. As various cloud users with different QoS requirement are migrating from an in-house data center to the public cloud, scheduling approaches for such dynamic environment should have the adaptability to adjust accordingly.

Multi-Objective Scheduling

Until now, optimizing multiple scheduling parameters, all variables except one are made constraints while transforming scheduling problem into Linear Program. Multi-objective scheduling has gained much attention in which more than one parameter is optimized simultaneously. This multi-objective optimization problem becomes more complex if conflicting criteria are optimized at the same time as time and energy. If completion time is to be minimized, the more powerful server is to be fired that consumes more energy. This multi-objective scheduling phenomenon is very common in cloud computing as objectives of cloud user; cloud providers can be independent. For example, cloud user can try to optimize time efficiency while at the same time the service provider wants to optimize resource utilization. Even for a single stakeholder some time, more than one scheduling parameter need to be optimized for example a cloud user may need to the optimized response time of its application with minimum budget.

Scheduling for Big Data

Over the past few years, the world has witnessed the vast generation of data from a variety of sources whether it is a scientific lab, e-commerce site, business enterprises, banking system, etc. This huge amount of data is characterized by 5 V's that is velocity, variety, volume, veracity, and value. There are many distributed framework like Hadoop MapReduce, Hadoop Spark, GraphLab, Microsoft Dryad [78] to process it efficiently. Now, everything is migrating towards the cloud;big data processing has made its entry to the cloud computing environment. In heterogeneous cloud environment scheduling of big data for processing through these frameworks imposes many challenges which need to be addressed.

5.1 Secure Scheduling

Data security, at all times, is one of the most important concerns to cloud users because their data may be seized or stolen by malicious parties during those data flows, specifically for less protected hybrid cloud systems. Research is required to implement scheduling in a way that it safeguards the sensitive and/or private information related to tasks/users. This type of scheduling is significant when the scheduled jobs carry private and/ or special information about various subjects in a given context.

6. CONCLUSION

In cloud computing, the problem of scheduling plays a significant role to optimize QoS requirements of multiple stake holders at the application as well as at virtualization layer. As the problem is NP-hard, the traditional scheduling algorithms fail to exhibit the required performance and take exponential time to produce the best schedules. On the other hand, the scheduling algorithms based on a scheme like *heuristic*, *meta-heuristic* and *hybrid heuristic*produces better results and are extensively used to schedule different categories of application (*e.g.*, set of independent tasks, scientific workflow, MapReduce jobs, etc.).

This survey paper discusses various cloud scheduling algorithms used only at the application layer. Algorithms are categorized on the basis of fundamental techniques used for design. Among these techniques, nature-inspired meta-heuristics are problem independent techniques which are used in many of scheduling algorithms. Whereas the heuristic techniques are problem dependent, hence, their scope is limited and specific to the environment for which they are designed. Both these techniques have their limitations which are overcome by hybrid-heuristic techniques. It takes the advantages of complementary benefits for heuristic and meta-heuristic techniques. During the study, we have covered many application types which are usually scheduled on cloud resources. Due to diverse characteristics of such applications, the design of scheduling algorithms differs in large scale. In the end, we provide the current research trends in this evolving area.

ACKNOWLEDGMENT

The authors would like to thank Ministry of Electronics and Information Technology, Government of India, and Digital India Corporation for providing financial assistance to this work under the Visvesvaraya Ph.D. scheme for Electronics and IT.

REFERENCES

1. T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," *2010 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, pp. 27–33, 2010.
2. D. Agarwal and S. Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment," *arXiv Prepr. arXiv1404.2076*, vol. 9, no. 7, pp. 344–349, 2014.
3. M. Armbrust, A. Fox, R. Griffith, A. Joseph, and RH, "Above the clouds: A Berkeley view of cloud computing," *Univ. California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009.
4. Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–33, 2015.
5. T. Kaur and I. Chana, "Energy Efficiency Techniques in Cloud Computing?:," vol. 48, no. 2, 2015.
6. A. V Vasilakos, "Cloud Computing?: Survey on Energy Efficiency," vol. 47, no. 2, 2014.
7. M. D. D. E. Assuncao and L. Lefevre, "A Survey on Techniques for Improving the Energy Efficiency of Large-Scale Distributed Systems," vol. 46, no. 4, 2014.
8. A. Hameed *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," 2014.
9. A. Bardsiri and S. Hashemi, "A Review of Workflow Scheduling in Cloud Computing Environment," *Int. J. Comput. Sci. Manag. Res.*, vol. 1, no. 3, pp. 348–351, 2012.
10. M. Masdari, S. ValiKardan, Z. Shahi, and S. I. Azar, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *Journal of Network and Computer Applications*, vol. 66. pp. 64–82, 2016.
11. M. G. P. C, M. K. P. K, P. Sheelvanthmath, and P. Ashwini, "A Survey on Energy Aware Scheduling of VMs in Cloud Data Centers," vol. 4, no. 3, pp. 226–229, 2015.
12. S. Singh and I. Chana, "QoS-Aware Autonomic Resource Management in Cloud Computing?:," vol. 48, no. 3, 2015.
13. M. Kalra and S. Singh, "REVIEW A review of metaheuristic scheduling techniques in cloud computing," *Egypt. Informatics J.*, vol. 16, no. 3, pp. 275–295, 2015.
14. Z. An, "Allocation of Virtual Machines in Cloud Data Centers — A Survey of Problem Models and Optimization Algorithms," vol. 48, no. 1, pp. 1–34, 2015.
15. H. Kaur and M. Singh, "Review of Various Scheduling Techniques in Cloud Computing," vol. 1, no. 2, pp. 30–36, 2012.
16. D. Du and P. Brucker, "Scheduling Algorithms." JSTOR, 2008.
17. T. H. Cormen, *Introduction to algorithms*. MIT press, 2009.
18. B. Saovapakhiran, G. Michailidis, and M. Devetsikiotis, "Aggregated-DAG scheduling for job flow maximization in heterogeneous cloud computing," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–6.
19. M. Rahman, X. Li, and H. Palit, "Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, 2011, pp. 966–974.
20. J. Yu and R. Buyya, "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints using Genetic Algorithms," *Sci. Program. J.*, vol. 14, no. 3, pp. 217–230, 2006.
21. F. Xhafa and A. Abraham, "Computational models and heuristic methods for Grid scheduling problems," *Futur. Gener. Comput. Syst.*, vol. 26, no. 4, pp. 608–621, 2010.
22. Y. Chawla and M. Bhonsle, "A study on scheduling methods in cloud computing," *Int. J. Emerg. Trends Technol. Comput. Sci.*, vol. 1, no. 3, pp. 12–17, 2012.
23. J. Y.-T. Leung, "A new algorithm for scheduling periodic, real-time tasks," *Algorithmica*, vol. 4, no. 1–4, pp. 209–219, 1989.
24. M. L. Dertouzos and A. K. Mok, "Multiprocessor online scheduling of hard-real-time tasks," *IEEE Trans. Softw. Eng.*, vol. 15, no. 12, pp. 1497–1506, 1989.
25. M. Wieczorek, A. Hoheisel, and R. Prodan, "Taxonomies of the Multi-Criteria Grid Workflow Scheduling Problem," *Grid Middlew. Serv.*, pp. 237–264, 2008.
26. M. Lachmann, S. Szamado, and C. T. Bergstrom, "Cost and conflict in animal signals and human language," *Proc. Natl. Acad. Sci.*, vol. 98, no. 23, pp. 13189–13194, Nov. 2001.
27. K. M. F. Elsayed and A. K. F. Khattab, "Channel-aware earliest deadline due fair scheduling for wireless multimedia networks," *Wirel. Pers. Commun.*, vol. 38, no. 2, pp. 233–252, Jul. 2006.
28. W. H. W. H. Kohler, *PRELIMINARY EVALUATION OF THE CRITICAL PATH METHOD FOR SCHEDULING TASKS ON MULTIPROCESSOR SYSTEMS.*, vol. C–24, no. 12. 1975, pp. 1235–1238.
29. R. G. Ingalls and D. J. Morrice, "PERT Scheduling With Resource Constraints Using Qualitative Simulation Graphs," *Proj. Manag. J.*, vol. 35, no. 3, pp. 5–14, 2004.
30. H.-J. Schütz and R. Kolisch, "Approximate dynamic programming for capacity allocation in the service industry," *Eur. J. Oper. Res.*, vol. 218, no. 1, pp. 239–250, 2012.
31. D. Shi and T. Chen, "Optimal periodic scheduling of sensor networks: A branch and bound approach," *Syst. Control Lett.*, vol. 62, no. 9, pp. 732–738, 2013.
32. P. Kumar and A. Verma, "Independent Task Scheduling in Cloud Computing by Improved Genetic Algorithm," *Int. J. Adv. Res. Comput. Sci. Softw. Eng. Res.*, vol. 2, no. 5, pp. 5–8, 2012.
33. G. Liu, J. Li, and J. Xu, "An improved min-min algorithm in cloud computing," in *Advances in Intelligent Systems and Computing*, 2013, vol. 191 AISC, pp. 47–52.
34. R. Singh and S. Singh, "Score Based Deadline Constrained Workflow Scheduling Algorithm for Cloud Systems," *Int. J. Cloud Comput. Serv. Archit.*, vol. 3, no. 6, pp. 31–41, 2013.
35. L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *J. Internet Serv. Appl.*, vol. 2, no. 3, pp. 207–227, Dec. 2011.
36. A. Verma and S. Kaushal, "Deadline and Budget Distribution based Cost- Time Optimization Workflow Scheduling Algorithm for Cloud," *Int. Conf. Recent Adv. Futur. Trends Inf. Technol.*, pp. 1–4, 2012.
37. D. Poola, K. Ramamohanarao, and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Procedia Comput. Sci.*, vol. 29, pp. 523–533, 2014.
38. K. Bessai, S. Youcef, A. Oulamara, C. Godart, and S. Nurcan, "Bi-criteria workflow tasks allocation and scheduling in cloud computing environments," *Proc. - 2012 IEEE 5th Int. Conf. Cloud Comput. CLOUD 2012*, pp. 638–645, 2012.
39. J. H. Abawajy, "Fault-tolerant scheduling policy for grid computing systems," *18th Int. Parallel Distrib. Process. Symp. 2004. Proceedings.*, vol. 00, no. C, pp. 0–6, 2004.
40. J. Wang, W. D. Bao, and X. M. Zhu, "Fault-tolerant scheduling algorithm for real-time tasks in virtualized cloud," *Tongxin Xuebao/Journal Commun.*, vol. 35, no. 10, pp. 2545–2558, 2014.
41. M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments," *OSDI '08 Proc. 8th USENIX Conf. Oper. Syst. Des. Implement.*, pp. 29–42, 2008.
42. Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, "SAMR: A Self-adaptive MapReduce Scheduling Algorithm in Heterogeneous Environment," in *2010 10th IEEE International Conference on Computer and Information Technology*, 2010, pp. 2736–2743.
43. X. Sun, C. He, and Y. Lu, "ESAMR: An enhanced self-

adaptive MapReduce Scheduling Algorithm," in *Proceedings of the International Conference on Parallel and Distributed Systems - ICPADS*, 2012, pp. 148–155.
44. S.-J. Yang and Y.-R. Chen, "Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds," *J. Netw. Comput. Appl.*, vol. 57, pp. 61–70, 2015.
45. F. Abazari, M. Analoui, H. Takabi, and S. Fu, "MOWS?: Multi-objective workflow scheduling in cloud computing based on heuristic algorithm," *Simul. Model. Pract. Theory*, vol. 93, no. October 2018, pp. 119–132, 2019.
46. J. Xue, L. Li, S. Zhao, and L. Jiao, "A Study of Task Scheduling Based On Differential Evolution Algorithm in Cloud Computing," vol. 9, no. 1, pp. 4–7, 2014.
47. C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing," *2009 5th Int. Conf. Wirel. Commun. Netw. Mob. Comput.*, pp. 1–4, 2009.
48. G. E. Junwei, "Research of cloud computing task scheduling algorithm based on improved genetic algorithm," vol. 1, no. Iccsee, pp. 2134–2137, 2013.
49. S. Banerjee, I. Mukherjee, and P. K. Mahanti, "Cloud Computing Initiative using Modified Ant Colony Framework," *Eng. Technol.*, vol. 56, no. 8, pp. 221–224, 2009.
50. H. Liu, D. Xu, and H. K. Miao, "Ant colony optimization based service flow scheduling with various QoS requirements in cloud computing," *Proc. - 1st ACIS Int. Symp. Softw. Netw. Eng. SSNE 2011*, pp. 53–58, 2011.
51. L. Zhu, Q. Li, and L. He, "Study on Cloud Computing Resource Scheduling Strategy Based on the Ant Colony Optimization Algorithm," *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 5, pp. 54–58, 2012.
52. K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization," *2012 UKSim 14th Int. Conf. Comput. Model. Simul.*, pp. 3–8, 2012.
53. J. Kennedy and R. Eberhart, "Particle swarm optimization," *Neural Networks, 1995. Proceedings., IEEE Int. Conf.*, vol. 4, pp. 1942–1948 vol.4, 1995.
54. S. Pandey, L. Wu, S. M. Guru, and 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*, 2010, pp. 400–407.
55. M. A. Rodriguez and R. Buyya, "Deadline Based Resource Provisioning and Scheduling Algorithm for Scientific Workflows on Clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 222–235, 2014.
56. L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput. J.*, vol. 13, no. 5, pp. 2292–2303, 2013.
57. Y. Wen, J. Liu, W. Dou, X. Xu, B. Cao, and J. Chen, "Scheduling workflows with privacy protection constraints for big data applications on cloud," *Futur. Gener. Comput. Syst.*, Mar. 2018.
58. S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, "Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds," *Futur. Gener. Comput. Syst.*, vol. 29, no. 1, pp. 158–169, 2013.
59. M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, 2011, p. 49.
60. B. Yagoubi and Y. Slimani, "Dynamic load balancing strategy for grid computing," *Trans. Eng. Comput. Technol.*, vol. 13, pp. 260–265, 2006.
61. X. Wen, M. Huang, and J. Shi, "Study on resources scheduling based on ACO allgorithm and PSO algorithm in cloud computing," *Proc. - 11th Int. Symp. Distrib. Comput. Appl. to Business, Eng. Sci. DCABES 2012*, vol. 1, no. 6, pp. 219–222, 2012.
62. A. G. Delavar and Y. Aryan, "A Goal-Oriented Workflow Scheduling in Heterogeneous Distributed Systems," *Int. J. Comput. Appl.*, vol. 52, no. 8, pp. 27–33, 2012.
63. S. Yassa, R. Chelouah, H. Kadima, and B. Granado, "Multi-Objective Approach for Energy-Aware Workflow Scheduling in Cloud Computing Environments," *Sci. World J.*, vol. 2013, p. 13, 2013.
64. M. Mezmaz *et al.*, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 11, pp. 1497–1508, 2011.
65. A. Ghorbannia Delavar and Y. Aryan, "HSGA: A hybrid heuristic algorithm for workflow scheduling in cloud systems," *Cluster Comput.*, vol. 17, no. 1, pp. 129–137, 2014.
66. M. Shojafar, S. Javanmardi, S. Abolfazli, and N. Cordeschi, "FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method," *Cluster Comput.*, vol. 18, no. 2, pp. 829–844, 2015.
67. P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *International Conference on the Practice and Theory of Automated Timetabling*, 2000, pp. 176–190.
68. C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A Hyper-Heuristic Scheduling Algorithm for Cloud," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, pp. 1–1, 2014.
69. X.-S. Yang and S. Deb, "Cuckoo Search via Levy Flights," 2010.
70. R. G. Babukarthik, R. Raju, and P. Dhavachelvan, "Energy-aware scheduling using Hybrid Algorithm for cloud computing," *2012 Third Int. Conf. Comput. Commun. Netw. Technol.*, no. July, pp. 1–6, 2012.
71. N. Jafari Navimipour and F. Sharifi Milani, "Task Scheduling in the Cloud Computing Based on the Cuckoo Search Algorithm," *Int. J. Model. Optim.*, vol. 5, no. 1, pp. 44–47, 2015.
72. Y. Wen, H. Xu, and J. Yang, "A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system," *Inf. Sci. (Ny).*, vol. 181, no. 3, pp. 567–581, 2011.
73. J. Li, M. Qiu, Z. Ming, G. Quan, X. Qin, and Z. Gu, "Online optimization for scheduling preemptable tasks on IaaS cloud systems," *J. Parallel Distrib. Comput.*, vol. 72, no. 5, pp. 666–677, 2012.
74. X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Futur. Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1124–1134, 2011.
75. J. Singh and H. Singh, "Efficient tasks scheduling for heterogeneous multiprocessor using genetic algorithm with node duplication," *Indian J. Comput. Sci. Eng.*, vol. 2, no. 3, pp. 402–410, 2011.
76. Y. C. Lee and A. Y. Zomaya, "Minimizing energy consumption for precedence-constrained applications using dynamic voltage scaling," in *Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on*, 2009, pp. 92–99.
77. S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu, and 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*, 2014, pp. 43–52.
78. M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks," *ACM SIGOPS Oper. Syst. Rev.*, pp. 59–72, 2007.