Journal of Circuits, Systems, and Computers
Vol. 28, No. 6 (2019) 1930006 (28 pages)
© World Scientific Publishing Company
DOI: 10.1142/S021812661930006X

A Review of Cost and Makespan-Aware Workflow Scheduling in Clouds[*]

Pingping Lu[†], Gongxuan Zhang[†], Zhaomeng Zhu[‡], Xiumin Zhou[†],
Jin Sun[†] and Junlong Zhou[†,§]

[†]*School of Computer Science and Engineering,*
Nanjing University of Science and Technology,
200 Xiaolingwei Street, Nanjing, Jiangsu 210094, P. R. China

[‡]*School of Computer Science and Engineering,*
Nanyang Technological University,
50 Nanyang Avenue 639798, Singapore
[§]*jlzhou@njust.edu.cn*

Received 2 March 2018
Accepted 2 July 2018
Published 8 August 2018

Scientific workflow is a common model to organize large scientific computations. It borrows the concept of workflow in business activities to manage the complicated processes in scientific computing automatically or semi-automatically. The workflow scheduling, which maps tasks in workflows to parallel computing resources, has been extensively studied over years. In recent years, with the rise of cloud computing as a new large-scale distributed computing model, it is of great significance to study workflow scheduling problem in the cloud. Compared with traditional distributed computing platforms, cloud platforms have unique characteristics such as the self-service resource management model and the pay-as-you-go billing model. Therefore, the workflow scheduling in cloud needs to be reconsidered. When scheduling workflows in clouds, the monetary cost and the makespan of the workflow executions are concerned with both the cloud service providers (CSPs) and the customers. In this paper, we study a series of cost-and-time-aware workflow scheduling algorithms in cloud environments, which aims to provide researchers with a choice of appropriate cloud workflow scheduling approaches in various scenarios. We conducted a broad review of different cloud workflow scheduling algorithms and categorized them based on their optimization objectives and constraints. Also, we discuss the possible future research direction of the clouds workflow scheduling.

Keywords: Cloud computing; scientific workflow; scheduling; cost; makespan.

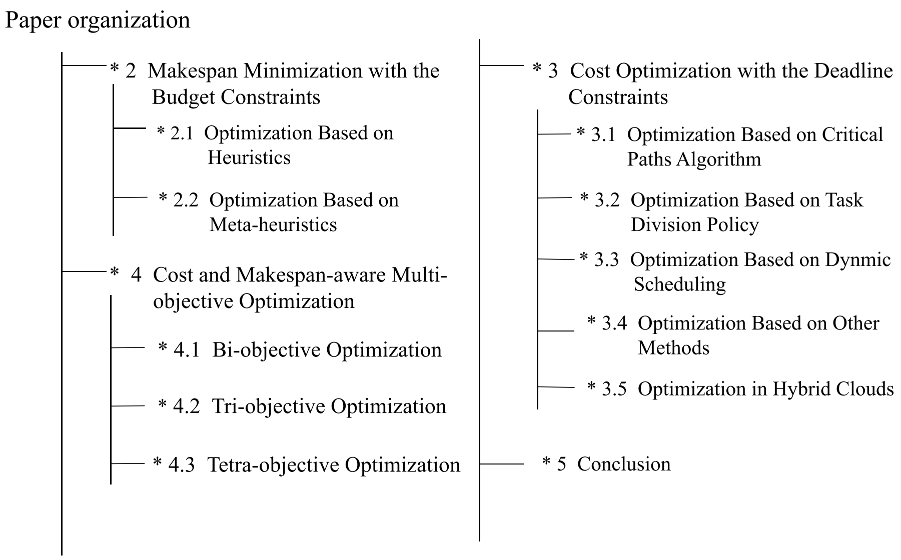
[*]This paper was recommended by Regional Editor Tongquan Wei.
[§]Corresponding author.

1. Introduction

The maturity of virtualization, high-speed networking and security technologies have made cloud computing flourish as a promising distributed computing technology for large-scale computations. Cloud service providers (CSPs) have tens of thousands of servers and storage devices that can provide cloud customers with ample computing and storage resources. The highly flexible availability of cloud resources have encouraging companies and organizations to migrate their computations to cloud platforms. Task scheduling is one of the major issues to achieve high performance in distributed systems such as Peer-to-Peer, Grid and cloud environment.[1] Moreover, cloud's powerful parallel computing power also makes researchers keen to study the task scheduling problem in the cloud environment.

Workflow is a common model to represent scientific computing including complex simulation and precise analysis of massive data. For example, CyberShake[2] based on PSHA (Probabilistic Seismic Hazard Analysis) is used to monitor earthquakes, and Montage[3] in the aerospace field synthesizes a series of FITS (Flexible Image Transport System) images into outer space nebula images. The executions of these workflows usually require a large amount of computing resources. Cloud computing can properly meet the resource requirement of these workflows. There have been a lot of studies in the workflow scheduling in environments like Grids and clusters. However, cloud platforms differ from traditional distributed platforms in their self-service resource management model and pay-as-you-go billing model. Thus, it is not easy to adapt the workflow scheduling algorithms designed for traditional platforms to present-day cloud platforms. In other words, the workflow scheduling problem needs to be reconsidered in cloud environment.

A workflow is usually formulated as Directed Acyclic Graphs (DAGs), where the nodes in the graphs represent the tasks in the workflow and the edges between the nodes represent the data or control dependencies between the tasks.[4] Workflow scheduling in cloud environments aims to achieve task allocation and scheduling in polynomial time. Workflow scheduling problem is widely considered to be a NP-complete problem.[5,6] These problems may not be able to find the optimal solutions in reasonable time when the problems are solved by mathematical analysis. Therefore, researchers generally use heuristics or meta-heuristics to optimize the NP-complete problem. When scheduling workflows in cloud, the makespan and monetary cost of the workflow executions are most concerned by both the cloud providers and customers. Makespan refers to the duration of the entire workflow from start to finish, and monetary cost is the cost that users need to pay to CSPs because of using cloud resources. The cloud customers usually want that workflow applications can be executed efficiently with minimized cost. For CSPs, the proper scheduling strategies that meet the requirements of customer' service quality can achieve greater profits and market competitiveness. Therefore, in this paper, we focus on time and/or



Paper organization
* 2 Makespan Minimization with the Budget Constraints
* 2.1 Optimization Based on Heuristics
* 2.2 Optimization Based on Meta-heuristics
* 4 Cost and Makespan-aware Multi-objective Optimization
* 4.1 Bi-objective Optimization
* 4.2 Tri-objective Optimization
* 4.3 Tetra-objective Optimization
* 3 Cost Optimization with the Deadline Constraints
* 3.1 Optimization Based on Critical Paths Algorithm
* 3.2 Optimization Based on Task Division Policy
* 3.3 Optimization Based on Dynmic Scheduling
* 3.4 Optimization Based on Other Methods
* 3.5 Optimization in Hybrid Clouds
* 5 Conclusion

Fig. 1. Organization of the paper.

cost-aware scheduling technologies of workflows in cloud environments. Unless otherwise stated, the workflow models studied in this paper are based on DAGs.

In this paper, we study more than 80 literatures related to the topic of cloud workflow scheduling. We classify them into three categories based on optimization objectives and constraints: (1) studies on the workflow scheduling to minimize the makespan within the given budget in cloud environment, (2) studies on the deadline-constrained cost-optimization workflow scheduling in clouds, and (3) studies on multi-objective workflow scheduling which optimize makespan and cost of workflow execution simultaneously.

The structure of this paper is shown in Fig. 1. In Sec. 2, we discuss the techniques and researches on scheduling budget-constrained workflows. In Sec. 3, we study

Table 1. Main abbreviations in the paper.

Abbreviation	Definition
DAGs	Directed Acyclic Graphs
CSPs	Cloud Service Providers
VM	Virtual Machine
QoS	Quality-of-Service
IaaS	Infrastructure as a Service
HV	Hyper-Volume
PSO	Particle Swarm Optimization
PCH	Path Clustering Heuristic
GA	Genetic Algorithm
HEFT	Heterogeneous Earliest Finish Time
ACO	Ant Colony Optimization

the works on scheduling deadline-constrained workflows. In Sec. 4, we summarize the multi-objective workflow scheduling problems and study related scheduling algorithm. Section 5 discusses further works and concludes the paper. To facilitate discussions, we summarize the abbreviations used in this paper as shown in Table 1.

2. Makespan Minimization Under Budget Constraints

In general, for real-time applications, the user wants to perform tasks within the budget constraint, especially some users like start-up companies, who want to complete workflow scheduling in a short time as much as possible, but they could only afford a tight budget. There have been many studies[7–21] discussing the problem of effectively scheduling workflows under given budget constraints. According to the classifications of the algorithms, we classify the scheduling algorithms proposed in these studies into heuristics[7–17] and meta-heuristics[18–21] which are briefly introduced in Tables 2 and 3, respectively.

2.1. *Optimization based on heuristics*

Heuristic algorithm is a class of algorithms based on intuitionism or experience. It can find out the feasible solution of the problem within a reasonable time and space. Although the deviation between the feasible solution and the optimal solution

Table 2. Brief introduction of heuristics.

Approach	References
ScaleStar and DeSlack policy	Zeng *et al.*[7]
Greedy policy	Wylie *et al.*[8]
	Lin and Wu[9]
	Wu and Cao[10]
	Wang and Shi[11]
Online-oriented strategy	Arabnejad and Barbosa[12]
Budget distribution strategy	Rodriguez and Buyya[13]
	Mohan *et al.*[14]
	Arabnejad *et al.*[15]
	Mao and Humphrey[16]
	Caron *et al.*[17]

Table 3. Brief introduction of meta-heuristics.

Approach	References
Particle swarm optimization algorithm	Wang *et al.*[18]
Genetic algorithm	Verma and Kaushal[19]
	Singh and Singh[20]
BAT	Kaur and Singh[21]

cannot be estimated, for some complicated optimization problems that classical mathematical methods cannot solve (such as the workflow optimization scheduling problem we discuss in this paper), the feasible solution is still valuable.

Zeng *et al.*[7] proposed a budget-conscious scheduling algorithm called ScaleStar, which minimizes the execution time of workflow under budget constraint. ScaleStar considers the requirements of users and CSPs. When executing workflow applications on clouds provides, users may expect that the monetary cost is brought by only the actually consumed computing resources. However, at this moment, cloud platforms usually charge customers on the basis of the activity time of virtual machines (VMs). Therefore, the authors proposed an adjustment policy, refer to as DeSlack, to minimize the number of idle VM times while not affecting the overall makespan of the schedule. This strategy makes ScaleStar can minimize the makespan of workflow execution and improve the utilization of resources under given budget constraint. However, ScaleStar needs to recalculate cost and execution time for every re-scheduling step which increases time complexity.

The greedy strategy is often used to solve the optimization problem, which is from the initial state of the problem through a number of greedy choices to hope to get the global optimal value of the problem. There have been some studies[8–11] that apply greedy strategy to the workflow scheduling in cloud environments. Wylie *et al.*[8] implemented modification to the Hadoop framework to support fully integrated workflow scheduling and proposed a greedy budget-constrained workflow scheduling algorithm on the Hadoop MapReduce platform. Lin and Wu[9] designed a heuristic algorithm called Critical-Greedy (CG). In this model, the time of the workflow only takes account of the execution time of the tasks on VM while the data transfer time between tasks is ignored. However, Wu and Cao[10] considered the data transfer, module execution, and I/O operations while using the greedy strategy to solve the workflow scheduling in cloud. Wang and Shi[11] considered budget-driven task-level scheduling algorithms for a batch of MapReduce jobs on a set of provisioned heterogeneous resources in cloud platforms. The authors organized the batch of MapReduce jobs as a k-stage workflow. The authors developed two greedy algorithms, global greedy budget (GGB) and gradual refinement (GR), with different greedy strategies. In GGB, the idea of the local greedy algorithm is extended to the efficient global distribution of the given budget with minimum makespan as a goal. At the same time, applying iteratively the DP algorithm to the distribution of exponentially reduced the budget so that the solutions are gradually refined in GR. The empirical results prove that the proposed optimal algorithms have the great efficiency for performance optimization of MapReduce workflow.

In recent years, some of the significant algorithms have been used for online scientific workflow scheduling, such as FDWS,[22] RANK-HYBD.[23] However, these algorithms do not take into account of cost, thus, Arabnejad and Barbosa[12] proposed a new strategy that considers the budget constraint and execution time and applies to RANK-HYBD and FDWS. This strategy gives a new priority rank for the task selection process that is the produce of two factors (time priority and cost priority) and a quality measure factor for the processor selection phase that is based on the combination of makespan and budget. The proposed strategies show better performances in almost all experimental cases.

The literatures[13–17] applies the budget distribution strategy for workflow scheduling in cloud. The strategy means that the workflow model is divided into different partitions according to a certain manner. In the same partition, the tasks are independent of each other. Among the different partitions, the tasks have at least one data dependencies. After creating the partitions, each partition is assigned to a sub-budget in a certain way. Rodriguez and Buyya[13] proposed a scheduling algorithm called BAGS. The BAGS includes a budget allocation strategy, which guides the individual expenditure of tasks and makes dynamic resource allocation and scheduling decisions to adapt to changes of the cloud environment. Mohan *et al.*[14] divided workflow into different partitions by using a topological manner. Moreover, a threshold value for each partition was given that provides more processing power for the tasks in each partition by borrowing some budget from the next partition. Arabnejad *et al.*[15] presented two budget division schemes, namely *Area* and *All in*, to efficiently handle data-intensive workflow scheduling. In *Area* strategy, height and width of a workflow are considered. In *All in* strategy, placing the entire budget on the entry level and any remainders are trickled down to later levels. Mao and Humphrey[16] presented two auto-scaling workflow scheduling algorithms, namely scheduling-first algorithm and scaling-first algorithm, to minimize the execution time under the budget constraints. The scheduling-first method allocates the given budget into each task of workflow according to the task priority and then arranges as many tasks as possible to the fastest execution VMs under the sub-budget constraint. Once the VM type of each task is determined, the auto-scaling mechanism will obtain the VM instances based on the scheduling plan. The scaling-first method firstly determines the type and the number of VMs under the budget constraints, and then schedules the tasks to the obtained resources based on the task priority to optimize the weighted average turnover time. In general, the proposed two methods make resource provisioning and task scheduling in different orders, and they show different advantages within different budget ranges. It is worth mentioning that Caron *et al.*[17] used the budget division strategy to deal with nondeterministic workflow model. Compared with DAGs workflow model, the proposed workflow model not only includes AND-join and AND-split structures (Fig. 2(a)) but also OR-joint, OR-split (Fig. 2(b)) and Cycle (Fig. 2(c)). The workflow model has certain complexity and research value but the resource assumption is limited in which the speed of CPU is same.

2.2. *Optimization based on meta-heuristics*

The meta-heuristic algorithm is a kind of generic heuristic algorithm. It is the combination of random search and local search. Compared with other heuristics,

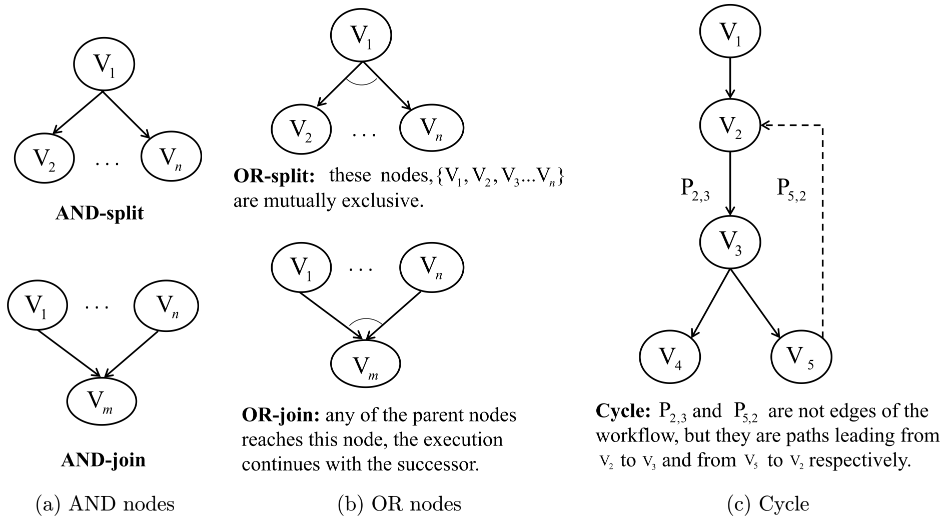

V1
V2
...
Vn
AND-split
V1
...
Vn
Vm
AND-join
(a) AND nodes
V1
V2
...
Vn
OR-split: these nodes, {V1, V2, V3...Vn} are mutually exclusive.
V1
...
Vn
Vm
OR-join: any of the parent nodes reaches this node, the execution continues with the successor.
(b) OR nodes
V1
V2
P2,3
P5,2
V3
V4
V5
Cycle: P2,3 and P5,2 are not edges of the workflow, but they are paths leading from V2 to V3 and from V5 to V2 respectively.
(c) Cycle

Fig. 2. Nondeterministic workflow control nodes and constructs.

meta-heuristics has the following two characteristics: (1) the optimization mechanism of meta-heuristic algorithm does not rely heavily on the organization structure information of the algorithm, and can be widely applied to the combination optimization problems and other science computing, and (2) random search is introduced to make the solutions more diverse. The classical meta-heuristic algorithm mainly imitates the natural body algorithm, mainly includes Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Ant Colony Optimization (ACO) and so on.

Wang *et al.*[18] proposed a novel evolutionary algorithm based on PSO to find the approximate optimal solution. In this algorithm, the process to derive an optimal task-to-VM mapping solution is abstracted as the process where particles automatically search and update the best particle through "mutual learning" until the iteration condition is not satisfied. Each of the particles represents a possible schedule. The search space in which particles search the optimal solution represents all possible scheduling solutions. The initialized particle swarm in each iteration can acquire a global best particle using the proposed new fitness function which applies to the problem to measure the quality of particles. After the iteration terminates, the particle representing the scheduling with the minimal makespan while satisfying the budget will be the final solution of the problem. Using real workflows to evaluate the proposed method, the results show that the approach can achieve preferable performance by increasing the number of particles and iterations.

GA is a specific class of meta-heuristic algorithms inspired by evolutionary biology. GA has become popular as a means of solving hard combinatorial optimization problems because of the good global optimization ability. Verma and

Kaushal[19] and Singh and Singh[20] have studied the budget-constrained makespan optimization problems for workflow scheduling in clouds based on GA. Singh and Singh extended GA to schedule workflow applications in unreliable cloud environment. The aim of the algorithm is to reduce the failure rate and optimize the makespan under the user's budget. Verma and Kaushal proposed Priority based Genetic Algorithm constrained the budget (BCHGA) to schedule workflows to cloud resources. In proposed BCHGA, each task of workflow applications is assigned priority using bottom level (b-level) and top level (t-level). To increase the population diversity, these priorities are then used to produce the initial population.

The quality of the workflow' output depends on whether the tasks are scheduled on reliable VMs or not, because VMs that have low failure rate can successfully execute tasks allocated to them. However, scheduling a task on more reliable machine incurs more cost. Similarly, time and cost are conflicting parameters that also need to be balanced and optimized. Thus, Kaur and Singh[21] applied a recently developed meta-heuristic method, the BAT algorithm, to solve the multi-objective problem of workflow scheduling in cloud environment. The BAT algorithm is inspired by the echolocation behavior of the bats and exhibits features like parameter control, frequency tuning and automatic zooming into the region where the optimal solution lies when applied to different optimization problems.[24–26] For the problem of scheduling workflows in clouds, a bat individual represents the task to virtual machine mapping for each task in the workflow. In extended BAT, the fitness function considering the reliability factor is proposed.

3. Cost Optimization with Deadline Constraints

For most applications, the cost is the primary factor affecting Quality-of-Service (QoS). However, some special applications, such as industrial, aerospace and other fields, are sensitive to deadline. Therefore, there are some studies[27–50] for minimizing cost while meeting the given deadline. We divide them into five categories based on the scheduling strategy or the scheduling environment. The first four subsections are based on the different scheduling strategies on public cloud, and the last is based on hybrid cloud, which are briefly introduced in Tables 4–8, respectively.

Table 4. Optimization based on critical paths algorithm.

Approach	References
Critical paths algorithm	Zhu *et al.*[27]
Deadline early tree algorithm	Yuan *et al.*[28]
Partial critical paths algorithm	Lin *et al.*[29]
	Arabnejad *et al.*[30] and Abrishami *et al.*[31]
Tasks replication strategy	Calheiros and Buyya[32]

Table 5. Optimization based on task division policy.

Approach	References
Partitioned balanced time scheduling	Byun *et al.*[33]
Task division policy	Kim *et al.*[34]
	Arabnejad *et al.*[35]
Clustering	Deldari *et al.*[36]

Table 6. Optimization based on dynamic scheduling strategy.

Approach	References
Just-in-time algorithm	Sahni and Vidyarthi[37]
Auto-scaling and online-oriented strategy	Mao and Humphrey[38]
GA	Meena *et al.*[39]

Table 7. Optimization based on other methods.

Approach	References
Greedy strategy	Kim *et al.*[40]
N/A	Kang *et al.*[41]
GA	Verma and Kaushal[42]
PSO	Rodriguez and Buyya[43]
Probabilistic list scheduling and ant colony optimization	Wu *et al.*[44]
Probabilistic scheduling	Zhou *et al.*[45]
Privacy-oriented	Sharif *et al.*[46]

Table 8. Brief introduction of optimization in hybrid clouds.

Approach	References
Path clustering heuristic algorithm	Bittencourt and Madeira[47]
N/A	Toosi *et al.*[48]
Task division policy	Chopra and Singh[50]

3.1. *Optimization based on critical paths algorithm*

Critical path refers to the longest logical path from input to output. The activity on the critical path is called the critical activity, and the earliest start time of the activity is equal to its latest start time. In this subsection, the critical path based workflow scheduling in clouds is designed to allocate sub-deadlines for critical activities to meet the minimum cost under deadline constraints.

Zhu *et al.*[27] presented a two-step workflow scheduling algorithm to minimize the cloud cost within a user-specified execution time. The proposed heuristic workflow scheduling approach consists of two steps. In the first step, the approach divides modules into different layers by topological sorting. It then assigns each module with a certain priority value based on its computational load and maps it to the node that yields the lowest partial end-to-end delay (EED) from the starting module to the current module. This module mapping process is repeated to reduce the total EED until a convergence point is reached. In the second step, the authors improve the resource utilization rate by reducing the overhead of VM's startup and shutdown times as well as the idle time. The modules may be mapped in such a way that some of them may share the same VM for reduced startup and shutdown overhead, or some VMs may be released early to save the idle time until the next active module arrives. The simulation results show that the approach consistently achieves lower computing overhead or higher resource utilization than existing methods within the execution time bound and also significantly reduces the total execution time by strategically selecting appropriate mapping nodes for prioritized modules.

An effective and efficient heuristic Deadline Early Tree (DET) approach is proposed by Yuan *et al.*[28] An early practicable schedule for a workflow application is defined as an Early Tree. According to the Early Tree, all tasks of the workflow are grouped and the Critical Path is given. In DET, tasks are partitioned into two types: critical and noncritical activities. For critical activities that are on the critical path, the optimal cost solution with the deadline constraint can be acquired by a dynamic programming strategy, and the whole deadline is segmented into some time windows based on the slack time float. For noncritical activities that are not on critical path, an iterative procedure is presented to maximize time windows while keeping the precedence constraints among activities. In terms of the time window assignments, a local optimization method is developed to minimize the execution cost. Experimental results show that the DET algorithm outperforms other recent leveling algorithms. In addition, the deadline division strategy of DET can be applied to all feasible deadlines.

Lin *et al.,*[29] Arabnejad *et al.*[30] and Abrishami *et al.*[31] adapted the Partial Critical Paths algorithm (PCP)[51] for workflow scheduling under the cloud environment. Lin *et al.*[29] proposed a scheduling strategy for scheduling workflow applications in multi-clouds. The approach takes into account some essential characteristics on Multi-Clouds such as various VM types, charge per time interval from different Cloud providers as well as homogeneous intra-bandwidth and heterogeneous inter-bandwidth. Compared to the traditional PCP, the new scheduling approach can compress and reduce the data transfer time along the partial critical path, which lead to a lower workflow cost. Arabnejad *et al.*[30] extended two algorithms, Proportional Deadline Constrained (PDC)[52] and Deadline Constrained Critical Path (DCCP).[53] The two extended algorithms refined their operation in task prioritization and back filling respectively. For task prioritization, the authors used eight different policies to analyze the impact of the execution order on the scheduling results and applied a set of new ranking polices for task prioritization. For back filling, three different policies (First Fit, Best Fit, Worst Fit) were adopted in DCCP.

Based on the previously proposed PCP algorithm applied to grid computing, Abrishami *et al.*[31] analyzed the difference between grid computing and cloud computing and extended the algorithm in the cloud environment. The extended two novel algorithms are a one-phase algorithm called IaaS Cloud Partial Critical Paths (IC-PCP) and a two-phase algorithm called IaaS Cloud Partial Critical Paths with Deadline Distribution (IC-PCPD2). IC-PCP has a similar policy to the deadline distribution phase of the original PCP, and the difference is that IC-PCP schedules each workflow task instead of specifying a sub-deadline. On the other hand, IC-PCPD2 is similar to the original PCP, but the deadline distribution phase and the planning phase are modified to adapt to the Clouds. The two proposed algorithms both have low time complexity that makes them fit for the large workflow.

The IC-PCP algorithm disregards deployment and boot time of VMs, by assuming that the earliest start time of the entry task is 0. However, VMs provisioned from CSPs are not immediately available for task execution; VMs need to be properly initialized and this time is not negligible. Therefore, Calheiros and Buyya[32] proposed an improved IC-PCP algorithm with task replication which uses idle time of provided resources and budget surplus to replicate tasks to reduce the effect of performance variation of cloud resources in the workflow execution time. The algorithm aims at minimizing the cost within deadline. Moreover, because the workflows are subject to a soft deadline, a bigger budget can be invested for execution of a workflow if it increases the likelihood of the deadline being met. The extra budget is expected to be proportional to the importance of the application to be completed by its deadline.

3.2. *Optimization based on task division policy*

The strategy of task division is to divide a workflow into different modules based on topological sorting. The tasks in each module are independent of each other, and there is at least one data dependency between adjacent modules. Then, the researchers implement parallel computation of tasks in each module by some specific strategies to achieve the optimization objectives.

Byun *et al.*[33] suggested a framework for the automatic executions of large-scale workflow applications on elastic computing resources provided clouds. PBTS (Partitioned Balanced Time Scheduling) as core part of the framework was extended based on the BTS[54,55] algorithm. It determines the optimized number of computing resources per charge unit in elastic computing environments, minimizing the total cost during the entire application lifetime and bridges the gap between cloud environments and the gap between workflow management system. Also, PBTS is able to handle workflows of some other structures which includes data-parallel, single, and even MPI-like tasks. The researchers are able to cloud while saving more cost with utility basis billing policy with the framework. However, in this work, the heterogeneity of cloud platforms are ignored. Kim *et al.*[34] designed and implemented a unified scientific cloud framework, called science gateway cloud (SGC). SGC is an agent between the user and the cloud provider which can effectively deal with various scientific applications on heterogeneous cloud resources. In the proposed framework, a cost-adaptive resource management scheme and a workflow scheduling scheme with the division policy to minimize cost while meeting the given deadline were proposed. The cost-adaptive resource management scheme with a virtual resource pool management policy is used to determine the amount of cloud resources with long term payment schemes[56] to save the cost of resource allocation. Also, the proposed workflow scheduling scheme is able to parallelize scientific applications. Through the simplification and efficient integration of the cost function model and the task affinity management of heterogeneous cloud services, the significant performance improvement of scientific applications is achieved.

Arabnejad *et al.*[35] presented a heuristic algorithm called the deadline distribution ratio (DDR) algorithm to solve workflow scheduling problem with the objective of minimizing the monetary cost under the deadline constraints. The workflow scheduling includes two significant phases: task selection phase and instance selection phase. The choice in these phases will naturally have a significant impact on the total cost of the scheduling plan while meeting the given deadline. One method that divides the total deadline of the workflow into a set of sub-deadlines makes it easy to handle the constraint satisfaction problem. Different deadline division strategies have an important impact on performance; the DDR thus focuses on solving the distribution of the deadline. In this case, the authors studied a series of deadline division strategies and designed a EWL strategy based on the combination of Width and Length strategies.

Deldari *et al.*[36] proposed a cluster combining algorithm (CCA) for workflow scheduling on multi-core resources in clouds. The CCA consists of two phases. In the pre-clustering phase, the workflow is divided into several primary clusters and a single-core processing resource for the execution of each cluster is leased. Then, in the combination-and-mapping phase, the best combination among the available clusters is chosen by a novel flexible scoring method. After that, the cluster tasks are mapped on multi-core processing resources by a step-by-step method. The proposed algorithm tries to reduce the execution time by leasing good resources with more cores (it means larger free time gaps will exist) when the deadline cannot be met. Once the given deadline is met, the proposed algorithm tries to reduce execution cost by filling free time gaps with tasks from other clusters.

3.3. *Optimization based on dynamic scheduling strategy*

The dynamic scheduling strategy mentioned in the subsection not only refers to online scheduling, but also includes resource aware and dynamic allocation. In this subsection, Sahni and Vidyarthi[37] and Mao and Humphrey[38] describe the online scheduling scheme of workflows, and Meena *et al.*[39] tell that the algorithm can adjust dynamically the scheduling scheme within a certain range of resource fluctuations.

Sahni and Vidyarthi proposed a dynamic cost-effective deadline-constrained heuristic algorithm called just-in-time (JIT-C) for scheduling a workflow in cloud environment. The proposed technique aims to utilize the advantages offered by cloud computing while taking into account the VM performance variability and instance acquisition delay to identify a just-in-time schedule of a deadline constrained workflow at lower costs. In the JIT-C, firstly, a pre-processing procedure combines pipeline tasks into a single task to maintain low execution cost. The objective of meeting the deadline then is achieved through continuous monitoring of the running tasks and dynamically making cost-effective scheduling decisions for subsequent tasks such that the deadline constraint is not violated. Further, it also exploits the available slack time with relaxed deadlines to generate cheaper schedules with lower execution costs. Performance evaluation on popular scientific workflows shows that the proposed algorithm can achieve better performance in comparison to several state-of-the-art heuristics.

Mao and Humphrey studied the auto-scaling problem for workflow applications that allow individual task deadlines and presented a method where the basic computing elements are the various sizes or costs of VMs, users specify performance requirements by assigning deadlines to tasks, and the objective is to minimize monetary cost while ensuring all tasks are finished within their deadlines. The proposed auto-scaling mechanism is based on a monitor control loop that accommodates dynamic changes such as delayed instance acquisitions and workload bursting. Moreover, the authors also used deadline assignment ideas to calculate an optimized resource scheme for each task and decide the number of instances using the load vector idea.

The existing researches for workflow scheduling on the clouds mainly focus on minimizing the makespan or minimizing the cost. However, most of them do not consider some important features of the cloud and the main issues that have a great impact on the scheduling plan, such as the performance variation and acquisition delays of VMs. To address the problem, a cost effective Genetic Algorithm (CEGA) has been proposed by Meena *et al.* to minimize the cost of workflow under the deadline. In CEGA, the authors developed a whole set of the operations of GA that include encoding, population initialization, crossover and mutation. Although the CEGA algorithm considers the main features of the cloud such as heterogeneity, on-demand resource provisioning and pay-as-you-go price model as well as some major issues such as VMs performance variation and booting time, it does not consider the cost of data transmission.

3.4. *Optimization based on other methods*

To guarantee Service Level Agreement (SLA) composed of the deadline and the given budget for workflow applications in mobile cloud, Kim *et al.*[40] proposed the two-phases algorithm with a cost adaptive VM management. Firstly, the greedy based workflow co-scheduling phase schedules a workflow by resource consolidation in a parallel manner to intensively reduce a cost with the deadline constraint. Secondly, the resource profiling based placement phase assigns a VM to a physical host in the multi-cloud using the profile on the property of clouds in order to comply with the budget while maximizing the service quality. Finally, the authors demonstrated that the proposed system with this two-phases algorithm outperforms other traditional systems in terms of both cost and processing time.

Currently, many CSPs such as Google, Amazon, adopt the resource pricing policy that is based on the fixed resource allocation time rather than the actual job processing time. In their strategy, VM allocation time is not a fine-grained time unit, but a coarse-grained time unit such as hour and day. That is to say, once the VM is occupied, the user needs to pay the cost until the end of the time unit, even if the VM is idle. To address the problem of unnecessary cost dissipation generated by idle VMs, Kang *et al.*[41] presented a heuristic based workflow scheduling scheme with the considerations of the real-world cloud-pricing model. The scheme includes two phases: VM packing phase and multi-requests to single resource (MRSR) phase. In VM packing phase, each task of the workflow already assigned with resource type is aggregated into the common VMs sequentially. That is to say, the fragmented partial instance hours is minimized since sub-tasks that require same resource type are aggregated to the generated VM. In MRSR phase, each sub-task that aggregated in VM packing phase is merged in parallel. Namely, tasks allocated in different VMs are merged into the single VM and processed concurrently. The proposed method achieves the significant cost saving within the user's SLA in terms of workflow deadline.

Verma and Kaushal[42] presented Deadline Constrained Heuristic based Genetic Algorithms (HGAs) to schedule workflow applications to cloud resources that minimize the execution cost while meeting the deadline for scheduling plan. In HGAs, each task of workflow is firstly assigned priority using bottom-level (b-level) and top-level (t-level). The b-level of a task is the length of the longest path from the task to the exit task (namely a leaf node in DAG). And the T-level of a task is defined as the length of the longest path from the task to the entry task (root node in DAG) without considering the execution time of task. Then, these priorities are used to create the initial population of BGA (B-level based GA), TGA (T-level based GA) and BTGA (B-level and T-level based GA) to increase the population diversity. Also, the different price models are considered in the proposed scheduling algorithm to extend well to different cloud environment.

Rodriguez and Buyya[43] presented a resource provisioning and scheduling strategy for scientific workflows on IaaS clouds and proposed a static algorithm based on PSO to minimize the cost within the given deadline. The proposed method considers the typical features of IaaS providers such as the heterogeneity and dynamic provisioning of unlimited computing resources as well as variation of VM performance.

To achieve this, resource allocation and scheduling are merged and modeled as an optimization problem. Then, PSO is used to solve this problem and generate a schedule that defines not only Task-to-VM mapping, but also the number and type of VMs that need to be rented, and the time that VMs need to be released or leased.

Wu *et al.*[44] proposed a heuristic Deadline-constrained Probabilistic List Scheduling (ProLis) as well as L-ACO to minimize execution cost of a workflow within the given deadline. The traditional list scheduling methods of optimizing makespan include two steps: construct an ordered list of tasks and assign each task from the list to the service. In the proposed ProLis algorithm, an additional step is also introduced that distributes the deadline to each task based on a definition of probabilistic upward rank. The L-ACO is based on Ant Colony Optimization Scheduling and ProLis, in which the ants first construct an ordered task list based on the pheromone trail and the probabilistic upward rank of a task and then use the same deadline distribution and service selection methods as ProLiS to find solutions. Moreover, the Min Max Ant System framework is utilized for updating the pheromone in L-ACO.

Zhou *et al.*[45] argued that Workflow-as-a-Service (WaaS) providers should have a sense of probability performance guarantees for a single workflow to expose the dynamics of the performance and cost of IaaS to the user. The authors developed a two-step workflow scheduling system called Dyna to minimize the monetary cost within the given probabilistic deadline guarantees. In the Dyna, the first step is to adopt an A*-based on-demand resource configuration method to select the resource type for each task of workflow. Second, starting with on-demand resource configuration, the authors used hybrid instance configuration refinement to consider the combination of on-demand and spot resource for executing tasks to further reduce cost. After the two optimization phases, the tasks of workflow are scheduled for execution on the cloud based on their hybrid resource configuration. The experimental results demonstrate that Dyna has the ability to meet the probabilistic deadline guarantees specified by the user and effectively reduce the cost at the same time.

Private data may be attacked when it can be accessed by the third-party cloud platforms. Therefore, Sharif *et al.*[46] proposed a method that minimizes the cost of scheduling and satisfies the deadline requirement and the privacy requirement of the workflow. The authors designed and implemented a multi-criteria workflow scheduler composed of MPHC-P1, MPHC-P2, and MPHC-P3 policies. The proposed scheduler includes two main modules: (1) the privacy protection module to meet the required privacy levels of all tasks and (2) scheduling module to obtain legal resources for workflow execution. Two privacy preserving policies (respectively BLP and Multi-terminal Cut) are applied in the privacy module, and three deadline awareness polices, namely, MPHC-P1, MPHC-P2, and MPHC-P3, are applied in the scheduling modules.

3.5. *Optimization in hybrid clouds*

The cloud platforms can be classified into public clouds, private clouds and hybrid clouds. The public clouds are managed by third-party cloud providers, and they usually charge users for their used computing resources. The private clouds are owned by users, and their resources can be freely used without charges. However, their scales are usually limited. The hybrid clouds are combinations of public clouds and privates. The hybrid clouds as a combination of public clouds and private clouds need to flexibly assign tasks to private cloud and public cloud based on the requirements of users. To solve the constrained scheduling problem in hybrid cloud, several algorithms have been proposed.

Bittencourt and Madeira[47] presented HCOC algorithm, which decides which resources should be leased from the public cloud and aggregated to the private cloud to provide sufficient processing power to execute a workflow within a given execution time. The HCOC algorithm consists of two main steps: (1) Make an initial scheduling using PCH [57] (Path Clustering Heuristic), a famous scheduling algorithm, in private cloud. (2) Select tasks to reschedule and selecting resource from public cloud to compose hybrid cloud by comparing three policies (Highest priority, Highest estimated finish time (EFT), Highest priority and EFT) when the makespan is larger than the deadline.

Toosi *et al.*[48] proposed a resource provisioning algorithm which can schedule data-intensive applications with deadline requirements in hybrid cloud environments. This algorithm computes the extra resources needed to complete application tasks within deadlines by considering some aspects such as network bandwidth, data locality, startup time of public cloud resources, and data transfer time. It is demonstrated that, compared to other existing approach, this algorithm can find the schedule which minimizes both the execution cost and the overall number of launched instances under strict deadlines for a sample data-intensive application. However, the authors only focused on scheduling and resource provisioning of Bag-of-Tasks applications (in a broad sense, they are also a kind of workflow) with a set of trivially parallel tasks which can be executed independent of one another.

Chopra and Singh[50] proposed a level workflow scheduling algorithm that decides which resources should be taken on lease from public cloud to complete the workflow tasks under deadline constraint. The proposed algorithm divides the entire workflow tasks into several levels, where each level of tasks is executed in parallel and the tasks are independent of each other. Task scheduling is accomplished at each level. The scheduling process of the proposed method has two phases, which schedule tasks to the private cloud and the public cloud, respectively. In private cloud scheduling phase, the method first tries to schedule the heaviest tasks that lead to high cost on private clouds since the private cloud resources are considered as free to use. In the second phase, tasks that cannot be executed on the private cloud are added to a list where a cost factor is assigned to each task for scheduling them to public cloud.

The results show that level-based hybrid scheduling algorithm is better than without level-based scheduling.

4. Cost and Makespan-Aware Multi-Objective Optimization

Although converting multi-objective optimization problems into single-objective optimization with constraints problems (such as Secs. 2 and 3) can effectively simplify the scheduling problem, this requires users to give constraints in advance. For many users, there may not be strict budget or time constraint requirements, and specifying constraints may result in scheduling failures or missing better solutions. Therefore, multi-objective optimization has become a hot research topic in recent years. This paper mainly focuses on the optimization of makespan and cost, but the QoS of users is also influenced by other factors, such as reliability and energy consumption. In this section, we divide the literatures[60–79] into three subsections based on the different optimization objectives. They are briefly introduced in Tables 9–11.

4.1. *Bi-objective optimization*

HEFT is the most famous scheduling algorithm for solving the performance-based workflow scheduling problem and is widely used in various existing workflow management systems, such as distributed embedded system[58,59] and the cloud platform we have discussed in this paper. Some researches[60–65] are based on the HEFT

Table 9. Some classifications of bi-objective optimization.

Approach	References
HEFT	Durillo *et al.*[60–62]
	Man and Huh[63]
	Su *et al.*[64]
	Li *et al.*[65]
PSO and HEFT	Verma and Kaushal[66]
Artificial bee colony	Udomkasemsub and Achalakul[67]
Event-driven and periodic rolling strategies	Chen *et al.*[68]
Task repetition strategy	Casas *et al.*[69]
Endocrine-based co-evolutionary multi-swarm algorithm	Yao *et al.*[70]
GA	Szabo and Kroeger[71]
	Zhu *et al.*[72]

Table 10. Some classifications of tri-objective optimization.

Approach	References
Service-level scheduling and meta-heuristics	Wu *et al.*[73]
Grey Wolf optimizer	Khalili and Babamir[74]
Service-oriented strategy	Tan *et al.*[75]

Table 11. Classifications of tetra-objective optimization.

Approach	References
Knee point driven evolutionary algorithm	Ye *et al.*[76]
GA	Fard *et al.*[77]
K-means based algorithm	Bousselmi *et al.*[78]
Priority based algorithm	Ji *et al.*[79]

algorithm to achieve the bi-objective workflow scheduling problem in cloud environment. The algorithm mainly includes two stages: In the first step, tasks are selected in descending order according to their priorities. In the second phase, the scheduler allocates the workflow tasks to VMs from the suitable set such that the total execution time and the monetary cost of running workflow are minimized.

Durillo *et al.*[60–62] achieved the optimization of cost and makespan of workflow in the cloud environment based on HEFT and got a flexible set of resource allocation schemes by using Pareto Technology. Durillo *et al.*[60,61] proposed a method called MOHEFT that extends HEFT. In the proposed scheduling framework, the price model follows the charge rule of Amazon EC2; it not only takes into account the computing cost but also the cost of data storage and data transfer. Another paper of Durillo *et al.*[62] also extended HEFT to handle multiple conflicting goals. Considering the different pricing models of different commercial cloud platforms, the authors proposed a universal pricing model with four parameters. Man and Huh[63] proposed a framework that combines computing resources on cloud and thick clients (TCs, such as PCs, tablets, etc.) residing on the local systems of computing components (CCs).[80] In the framework, TCs can handle tasks assigned to them and store a specific amount of user data when necessary to perform workflows. Resource proxy, a resource management component of the framework, is responsible for receiving and solving the scheduling problem of workflow submitted by CCs' side users. The key part of the framework is a scheduling algorithm, Cost with Finish Time-based (CwFT), based on HEFT. In CwFT, a novel utility function, which is based on performance of scheduling scheme and resource price, is used to select the best processing unit of each task. Su *et al.*[64] constructed a scheduling optimization problem with multiple VMs and different pricing models. The authors then designed a cost-efficient scheduling algorithm based on HEFT to better match the pricing model and the opacity of the cloud. The key component of the proposed algorithm is two scheduling heuristic strategies. The first heuristic method uses Pareto dominance to create a cost-effective schedule according to the task execution time and charges of the VMs. The second heuristic method complements the first heuristics, which attempts to minimize the monetary cost of noncritical tasks by extending the execution time. Li *et al.*[65] focused on the use of cloud resources for assigning large graph processing tasks and designed a framework called EComer that is easy to integrate into the existing cloud infrastructure. The core component of EComer is a cost-conscious scheduling heuristic

(CCSH) based on HEFT. The CCSH first constructs a task priority list and then maps the task with the highest priority value to the cost-efficient VM in cloud. The cost-awareness factor in the primary mapping stage effectively balances the trade-off between execution time and cost. The results show that CCSH outperforms HEFT by displaying significant monetary cost savings at a reasonable increase in overall execution time. However, what is not enough is that only one VM type and one pricing model is considered.

Verma and Kaushal[66] presented the HPSO (Hybrid Particle Swarm Optimization) algorithm based on nondominance sort to deal with the cloud workflow scheduling problem with multiple optimized objective functions. The proposed algorithm is a combination of a multi-objective PSO algorithm and a list based heuristic algorithm called BDHEFT[81] that was previously proposed by Verma and Kaushal The BDHEFT algorithm gives a trade-off schedule scheme between makespan and cost depending upon the user preference, and a multi-objective PSO algorithm produces a great range of potential solutions for multi-objective cloud workflow scheduling problem, so the hybrid of PSO and BDHEFT algorithm leads to more efficient behavior of the HPSO algorithm to schedule workflow applications on IaaS cloud.

Udomkasemsub and Achalakul[67] proposed a workflow scheduling framework that can efficiently schedule workflow with multiple objectives on a cloud system. The presented framework aims to handle complex workflows for data analysis by dealing with various workflow structures, short-term and long-term scheduling, dependency mapping on a heterogeneous cloud environment and so on. In addition, the framework also allows multiple objectives and constraints to be set. The core of the designed framework is the application of Artificial Bee Colony (ABC) to workflow scheduling. The ABC algorithm is one of swarm-based algorithms for combinatorial optimization problem that is inspired by the foraging behavior of honeybee colonies. It consists of three types of bees, respectively, onlooker bee, employed bee and scout bee. The food source location represents a solution of the problem, and the amount of nectar from the food source represents the quality of the relevant solution. ABC repeats iterations until a predefined stopping condition is satisfied.

Due to the dynamic and uncertain nature of the cloud environment, a large number of schedule interruptions (such as arrival of new workflows, variation of task execution time) may occur and the pre-scheduling plan may not be strictly enforced or effective as expected in actual execution. However, the most of researches ignored these uncertain factors, which may lead to a large difference between the expected behavior and the real execution behavior. To cover the issue, Chen *et al.*[68] designed an uncertainty-aware scheduling framework to reduce the influence of uncertain factors on workflow scheduling. Based on the framework, the dynamic workflows scheduling algorithm, combining event driven and periodic rolling strategies (EDRPS), was proposed that aims at reducing the economic cost and maximizing resource utilization while meeting the deadline requirements. In addition, the experiment implemented by using real-world workflows show that the proposed approach has better performance.

Most existing workflow scheduling methods in clouds generate scheduling plans based on a static number of resources. A few allowed users to execute workflow applications using different resource pool sizes. To overcome the limitations, Casas *et al.*[69] designed and implemented a Balanced and file Reuse-Replication workflow scheduling (BaRRs) algorithm for cloud environment to achieve the trade-off of execution time and money cost. BaRRS is based on three techniques and an in-depth workflow analysis. The proposed three techniques include queue balancing technique, file reuse technique and file replication technique. The BaRRs firstly uses queue balancing technique to divide a scientific workflow into multiple sub-workflows to balance resources utilization via parallelization. It also exploits file reuse and replication techniques to optimize the size of data which needs to be transferred between tasks at runtime. BaRRS performs a trade-off analysis to select the optimal solution based on execution time and monetary cost of scientific workflow and the experiments show its superior performance.

Considering multiple scheduling conflicting objectives, such as the total execution time (makespan), economic cost, and energy consumption, an endocrine-based co-evolutionary multi-swarm algorithm for multi-objective optimization problem (ECMSMOO) is proposed by Yao *et al.*[70] In ECMSMOO, multiple swarms are used and each of which uses an improved multi-objective particle swarm optimization to find nondominated solutions with one objective. In order to avoid falling into the local optimal solution of the traditional heuristic algorithm, an endocrine heuristic mechanism is embedded in the particle evolution process. Based on the strategy, an improved particle velocity update scheme is proposed. In the scheme, the update of particle velocity in each iteration is not only affected by the personal best, the swarm best, and the global best, but also the information of the particle's neighbor. In addition, the technology of competition and cooperation among swarms is also designed in ECMSMOO. Furthermore, to avoid the impact of elastic available resources, a management server was used to collect the available resources for scheduling. All of these strategies effectively improve ECMSMOO performance.

Szabo and Kroeger[71] and Zhu *et al.*[72] solve workflow scheduling in clouds based on GA. Szabo and Kroeger[71] proposed an optimization framework for task allocation in public cloud that considers some significant parameters, such as communication overhead, workflow runtime, and total execution cost. The proposed multi-objective optimization framework utilizes GA to determine the best number of cloud instances to be used. In the phase of encoding of chromosomes, the authors used two chromosomes to represent a solution of the scheduling problem (The first chromosome is an allocation strategy and maps workflow tasks to VMs, the other one represents a ordering of all tasks) and designed a set of problem-specific crossover and mutation operators. Zhu *et al.*[72] first analyzed the challenges of workflow scheduling under cloud environment compared with other distributed computing. Then, according to the characteristics of multi-objective optimization in cloud environment, an evolutionary algorithm is proposed which aims to optimize makespan and cost synchronously. In the proposed scheme, three methods of initial population are proposed to reduce the search space, and a three-dimensional chromosome encoding scheme reduces the length of chromosome encoding, that is, reduce the complexity of the algorithm. In addition, the new fitness function and the new genetic operators are also presented. All these works make the proposed algorithm show good performance in real workflow applications.

4.2. *Tri-objective optimization*

For workflow scheduling in the cloud environment, users want to minimize the makespan and the cost for execution workflow while the cloud providers want to maximize resource utilization to achieve profit maximization. A few researches consider both the requirements of users and cloud providers. To address the conflicting objectives, Wu *et al.*[73] and Khalili and Babamir[74] proposed respectively a market-oriented hierarchical workflow scheduling strategy in clouds. Wu *et al.* proposed a scheduling approach that mainly consists of two levels, namely service-level scheduling and task-level scheduling. Specifically, the service-level scheduling handles the Task-to-VM assignment where tasks of individual workflow application are mapped to cloud resources in the global cloud markets based on QoS requirements. The task-level scheduling handles the optimization of the Task-to-Service assignment in local cloud data centres in which the overall running cost of cloud workflow systems will be minimized under constraints. Khalili and Babamir extended the single objective heuristic algorithm GWO (Grey Wolf Optimizer) to optimize the makespan and cost as well as resource utilization. GWO is one of the swarm intelligence algorithms inspired from the hunting behavior of wolves. In GWO method, each wolf as a problem solution in search space has a position vector. The fitness function is used to evaluate the position of wolves. During the hunting process as the optimizing process, wolves update their position according to the three best values of fitness. Compared with other well-known meta-heuristic algorithms such as GA and PSO, GWO shows more effectiveness in finding optimal solutions; at the same time, it can avoid dispersion of the solution space and converge to the optimal positions. On the basis of GWO, the authors proposed PGWO in which optimal Pareto approach is used to address multi-objective problems. In the proposed PGWO method, the fitness was inspired by the SPEA2 algorithm and an external archive that keeps the nondominated solutions. To evaluate the performance of the PGWO, the authors implemented the extended method using the WorkflowSim. And the results show PGWO outperforms SPEA2.

In cloud environment, workflow scheduling is up against the threats of the inherent uncertainty and unreliability in the workflow applications. Thus, Tan *et al.*[75] proposed three-objective optimization problem that not only considers execution time and cost but also introduce a trust factor to meet the security requirements of private data. As a result, a trust-oriented workflow scheduling algorithm is proposed. The authors used Max-Min as the operator to model the trust evaluation function, the execution cost function, the execution time function. At the same time, a trust metric that includes direct trust and recommendation trust is used in the trust evaluation function. The experimental results show that the approach is effective and feasible.

4.3. *Tetra-objective optimization*

Ye *et al.*[76] presented a new workflow scheduling model that considers four objectives simultaneously: minimizing makespan, minimizing economic cost of the workflow execution, minimizing the average execution time for all workflow instances, and maximizing the reliability. In order to solve the four-objective scheduling problem, an improved knee-point-driven evolutionary (IKnEA) algorithm is proposed. KnEA[82] is a kind of evolutionary algorithm for solving many-objective optimization problems (MOPs) in recent years, which has been proved to be capable of outperforming several state-of-the-art many-objective optimization evolutionary algorithms for MOPs in terms of the two widely used performance indicators, namely, inverted generational distance (IGD) and hyper-volume (HV). Deficiently, KnEA may select solutions that have a larger distance from the hyperplane with less contribution to the increment in HV. In order to improve the performance of KnEA, the authors proposed two policies to improve the selection process in KnEA by introducing TOPSIS[83] and entropy weight.[84] Experimental results show that the IKnEA displays better performance in the majority of cases compared with KnEA.

Fard *et al.*[77] designed a general framework and proposed a list-based multi-objective heuristic scheduling algorithm (MOLS) for static workflow scheduling in heterogeneous computing environments such as grid and cloud. The MOLS algorithm tries to find out a solution that dominates the constraint vector or a solution which converges to the constraint vector and mainly consists of three phases. In the first phase, it divides the constraint vectors for all workflow activities based on the workload; that is to say, it estimates the objectives sub-constraints for each individual task using the constraint vector. In the second phase, it assigns a rank to each task of the workflow and sorts them in ascending order. And in the third phase, it tries to allocate the most appropriate resource to each task within the estimated sub-constraints. In addition, the authors analyzed and classified different objectives with respect to their impact on the optimization process and presented a four-objective case study comprising makespan, economic cost, energy consumption, and reliability.

Bousselmi *et al.*[78] proposed a dynamically reconfigurable framework for the deployment of large-scale scientific workflow (DR-SWDF) in the clouds that supports customizing the workflow deployment process based on a set of objectives and constraints of users or cloud service providers defined. In the proposed framework, the authors presented a K-means-based algorithm that supports dynamically clustering the workflows of sub-workflows to determine the most convenient approaches to be applied for their deployment and scheduling in clouds. Moreover, according to the input parameters of the workflow, the requirements of users, and the objectives of the cloud providers, the DR-SWDF also allows the personalized deployment of the workflow at a fine-grained level, such as the deployment of specific tasks of the workflow, using different approaches at runtime. The paper used makespan, economic cost, resources availability and reliability as the optimistic objectives, but the framework is generic and not limited to the above objectives.

In general, economic cost is the most important factor for providing cloud services. But the objectives may change due to some reasons such as time, which is the most important factor in emergency. In this case, it is impractical to reset the scheduling mechanisms only for an occasional incident. To address the problem, an adaptive scheduling plan is needed to solve scheduling problem in different conditions. Ji *et al.*[79] converted the problem to a multi-objective workflow scheduling with variable objective weights and presented a two-phase adaptive priority-based workflow scheduling algorithm (DRAWS) that considers the hottest four optimization objectives, namely makespan, cost, energy consumption and reliability. In the first phase of the DRAWS, the authors converted the workflow to a task list with task priority that is decided by the topology features of the workflow. In the second phase, according to the input which is the individual preference on different objectives, task priority was adjusted. Then tasks were assigned on suitable VMs.

5. Conclusion

The cost optimization is an important challenge for the workflow scheduling and resource management in clouds. In this work, we summarize different kinds of cost-aware scheduling algorithms and schemes and classify them according to different criteria. This work provides a rich and reliable information for future researchers. For the future trend of development, we believe that the multi-objective scheduling (especially to optimize reliability, resource utilization rate, time and cost) is very meaningful research. However, the multi-objective scheduling algorithms usually suffer from their high computation complexity, which requires researchers to realize the suitable combination of heuristic and meta-heuristic algorithms to achieve low complexity scheduling. On the other hand, at present, most of the studies are carried out for the off-line scheduling workflow, but for the actual operating environment, workflow should be allowed to be dynamic; therefore, we also thinks that one of the most important researches is online workflow scheduling in the future.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant Nos. 61272420 and 61502234) and the Natural Science Foundation of Jiangsu Province of China (Grant No. BK20150785). We are very grateful to the editor and reviewers for their valuable suggestions to improve the quality of this paper.

References

1. C. Li and L. Li, Hierarchical scheduling optimization scheme in hybrid cloud computing environments, *J. Circuits Syst. Comput.* **34** (2015) 15–59.
2. R. Graves, T. H. Jordan and S. Callaghan, CyberShake: A physics-based seismic hazard model for Southern California, *Pure Appl. Geophys.* **168** (2011) 367–381.
3. H. X. Wang and Y. Zhang, Scheduling method for multi-mode cloud workflows with deadlines, *J. Chinese Comput. Syst.* **37** (2016) 2437–2442.
4. G. Xie, Y. Chen, Y. Liu, Y. Wei, R. Li and K. Li, Resource consumption cost minimization of reliable parallel applications on heterogeneous embedded systems, *IEEE Trans. Ind. Inform.* **13** (2017) 1629–1640.
5. H. Topcuoglu, S. Hariri and M. Y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing, *IEEE Trans. Parallel Distrib. Syst.* **13** (2002) 260–274.
6. J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan and Y. Ma, Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms, *J. Syst. Softw.* **133** (2017) 1–16.
7. L. Zeng, B. Veeravalli and X. Li, ScaleStar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud, *2012 IEEE 26th Int. Conf. Advanced Information Networking and Applications*, Vol. 11, Fukuoka, Japan, 2012, pp. 534–541.
8. A. Wylie, S. Wei, J. P. Corriveau and W. Yang, A scheduling algorithm for hadoop mapreduce workflows with budget constraints in the heterogeneous cloud, *Int. Parallel and Distributed Processing Symp. (IPDPS)*, 2016, pp. 1433–1442.
9. X. Lin and C. Q. Wu, On scientific workflow scheduling in clouds under budget constraint, *2013 42nd Int. Conf. Parallel Processing*, Vol. 46, Lyon, France, 2013, pp. 90–99.
10. C. Q. Wu and H. Cao, Optimizing the performance of big data workflows in multi-cloud environments under budget constraint, *IEEE Int. Conf. Services Computing*, San Francisco, USA, 2016, pp. 138–145.
11. Y. Wang and W. Shi, Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds, *IEEE Trans. Cloud Comput.* **2** (2014) 306–319.
12. H. Arabnejad and J. G. Barbosa, Budget constrained scheduling strategies for on-line workflow applications, *Procedia Comput. Sci.* **8584** (2014) 532–545.
13. M. A. Rodriguez and R. Buyya, Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods, *ACM Trans. Auton. Adapt. Syst.* **12** (2017) 138–145.
14. A. Mohan, M. Ebrahimi, S. Lu and A. Kotov, Scheduling big data workflows in the cloud under budget constraints, *IEEE Int. Conf. Big Data*, 2017, pp. 2775–2784.
15. V. Arabnejad, K. Bubendorfer and B. Ng, A budget-aware algorithm for scheduling scientific workflows in cloud, *IEEE Int. Conf. High Performance Computing and Communications*, 2017, pp. 1188–1195.

16. M. Mao and M. Humphrey, Scaling and scheduling to maximize application performance within budget constraints in cloud workflows, *Int. Parallel and Distributed Processing Symp. (IPDPS)*, 2013, pp. 67–78.
17. E. Caron, F. Desprez, A. Muresan and F. Suter, Budget constrained resource allocation for non-deterministic workflows on an IaaS cloud, *Int. Conf. Algorithms and Architectures for Parallel Processing*, 2012, pp. 186–201.
18. X. Wang, B. Cao, C. Hou, L. Xiong and J. Fan, Scheduling budget constrained cloud workflows with particle swarm optimization, *IEEE Collaboration and Internet Computing*, Pennsylvania, United States, 2016, pp. 219–226.
19. A. Verma and S. Kaushal, Budget constrained priority based genetic algorithm for workflow scheduling in cloud, *Computing & Communication*, 2013, pp. 216–222.
20. L. Singh and S. Singh, A genetic algorithm for scheduling workflow applications in unreliable cloud environment, *Int. Conf. Security in Computer Networks and Distributed Systems*, 2014, pp. 139–150.
21. N. Kaur and S. Singh, A budget-constrained time and reliability optimization BAT algorithm for scheduling workflow applications in clouds (Elsevier Science Publishers B. V., Amsterdam, Netherlands, 2016), pp. 199–204.
22. H. Arabnejad and J. Barbosa, Fairness resource sharing for dynamic workflow scheduling on heterogeneous systems, *IEEE Int. Symp. Parallel and Distributed Processing with Applications*, 2012, pp. 633–639.
23. Z. Yu and W. Shi, A planner-guided scheduling strategy for multiple workflow applications, *Int. Conf. Parallel Processing-Workshops IEEE Computer Society*, 2008, pp. 1–8.
24. P. W. Tsai, J. S. Pan, B. Y. Liao, M. J. Tsai and V. Istanda, Bat algorithm inspired algorithm for solving numerical optimization problems, *Appl. Mech. Mater.* **148–149** (2012) 134–137.
25. X. Yang, Bat algorithm for multi-objective optimisation, *Int. J. Bio-Inspired Comput.* **3** (2011) 267–274.
26. B. Ramesh, V. Mohan and V. Reddy, Application of bat algorithm for combined economic load and emission dispatch, *Int. J. Electr. Eng. Telecommun.* **2** (2013) 1–9.
27. M. Zhu, Q. Wu and Y. Zhao, A cost-effective scheduling algorithm for scientific workflows in clouds, *International Performance Computing and Communications Conf.*, Vol. 326, 2013, pp. 256–265.
28. Y. Yuan, X. Li, Q. Wang and X. Zhu, Deadline division-based heuristic for cost optimization in workflow scheduling, *Inf. Sci.* **179** (2009) 2562–2575.
29. B. Lin, W. Guo, G. Chen, N. Xiong and R. Li, Cost-driven scheduling for deadline-constrained workflow on multi-clouds, *IEEE International Parallel and Distributed Processing Symp. Workshop*, 2015, pp. 1191–1198.
30. V. Arabnejad, K. Bubendorfer and B. Ng, Scheduling deadline constrained scientific workflows on dynamically provisioned cloud resources, *Future Gener. Comput. Syst.* **75** (2017) 348–364.
31. S. Abrishami, M. Naghibzadeh and D. H. J. Epema, Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds, *Future Gener. Comput. Syst.* **29** (2013) 158–169.
32. R. N. Calheiros and R. Buyya, Meeting deadlines of scientific workflows in public clouds with tasks replication, *IEEE Trans. Parallel Distrib. Syst.* **25** (2014) 1787–1796.
33. E. K. Byun, Y. S. Kee, J. S. Kim and S. Maeng, Cost optimized provisioning of elastic resources for application workflows, *Future Gener. Comput. Systems* **27** (2011) 1011–1026.

34. S. H. Kim, D. K. Kang, W. J. Kim, M. Chen and C. H. Youn, A science gateway cloud with cost-adaptive VM management for computational science and applications, *IEEE Syst. J.* **99** (2017) 1–13.
35. V. Arabnejad, K. Bubendorfer and B. Ng, Deadline distribution strategies for scientific workflow scheduling in commercial clouds, *Int. Conf. Utility and Cloud Computing*, 2017, pp. 70–78.
36. A. Deldari, M. Naghibzadeh and S. Abrishami, CCA: A deadline-constrained workflow scheduling algorithm for multicore resources on the cloud, *J. Supercomput.* **73** (2016) 1–26.
37. J. Sahni and D. Vidyarthi, A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment, *IEEE Transactions on Cloud Computing* **6** (2018) 2–18.
38. M. Mao and M. Humphrey, Auto-scaling to minimize cost and meet application deadlines in cloud workflows, *Int. Conf. High Performance Computing, Networking, Storage and Analysis*, 2011, pp. 1–12.
39. J. Meena, M. Kumar and M. Vardhan, Cost effective genetic algorithm for workflow scheduling in cloud under deadline constraint, *IEEE Access* **4** (2016) 5065–5082.
40. W. J. Kim, D. K. Kang, S. H. Kim and C. H. Youn, Cost adaptive VM management for scientific workflow application in mobile cloud, *Mobile Netw. Appl.* **20** (2015) 328–336.
41. D. K. Kang, S. H. Kim, C. H. Youn and M. Chen, Cost adaptive workflow scheduling in cloud computing, *Int. Conf. Ubiquitous Information Management and Communication*, 2014, pp. 1–8.
42. A. Verma and S. Kaushal, Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud, *Int. J. Grid Util. Comput.* **5** (2014) 96–106.
43. M. A. Rodriguez and R. Buyya, Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, *IEEE Trans. Cloud Comput.* **2** (2014) 222–235.
44. Q. Wu, F. IshiKawa, Q. Zhu, Y. Xia and J. Wen, Deadline-constrained cost optimization approaches for workflow scheduling in clouds, *IEEE Trans. Parallel Distrib. Syst.* **28** (2017) 3401–3412.
45. A. C. Zhou, B. He and C. Liu, Monetary cost optimizations for hosting workflow-as-a-service in IaaS clouds, *IEEE Trans. Cloud Comput.* **4** (2016) 34–48.
46. S. Sharif, P. Watson, J. Taheri, S. Nepal and A. Y. Zomaya, Privacy-aware scheduling SaaS in high performance computing environments, *IEEE Trans. Parallel Distrib. Syst.* **28** (2017) 1176–1188.
47. L. F. Bittencourt and E. R. M. Madeira, HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds, *J. Internet Serv. Appl.* **2** (2011) 207–227.
48. A. N. Toosi, R. Sinnott and R. Buyya, Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using aneka, *Future Gener. Comput. Syst.* **79** (2017) 765–775.
49. T. Wei, J. Zhou, K. Cao *et al.*, Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs, *IEEE Trans. Comput.-Aided Des. Integr. Circuits and Syst.*, Vol. 99 (2017), p. 1.
50. N. Chopra and S. Singh, Deadline and cost based workflow scheduling in hybrid cloud, *Int. Conf. Advances in Computing, Communications and Informatics*, 2013, pp. 840–846.
51. S. Abrishami, M. Naghibzadeh and D. H. J. Epema, Cost-driven scheduling of grid workflows using partial critical paths, *IEEE Trans. Parallel Distrib. Syst.* **23** (2012) 1400–1414.

52. V. Arabnejad and K. Bubendorfer, Cost effective and deadline constrained scientific workflow scheduling for commercial clouds, *Proc. 14th IEEE Int. Symp. Network Computing and Applications*, 2016, pp. 106–113.
53. V. Arabnejad, K. Bubendorfer, B. Ng and K. Chard, A deadline constrained critical path heuristic for cost-effectively scheduling workflows, *Proc. 8th IEEE Int. Conf. Utility and Cloud Computing*, 2015, pp. 242–250.
54. E. K. Byun, Y. S. Kee, E. Deelman, K. Vahi, G. Mehta and J. S. Kim, Estimating resource needs for time-constrained workflows, *Proc. 4th IEEE Int. Conf. e-Science*, 2008, pp. 31–38.
55. E. K. Byun, Y. S. Kee, J. S. Kim, E. Deelman and S. Maeng, Bts: Resource capacity estimate for time-targeted science workflows, *J. Parallel Distrib. Comput.* **71** (2011) 848–862.
56. S. Chaisiri, B. S. Lee and D. Niyato, Optimization of resource provisioning cost in cloud computing, *IEEE Trans. Serv. Comput.* **5** (2012) 164–177.
57. L. F. Bittencourt, C. R. Senna and E. R. M. Madeira, Enabling execution of service workflows in grid/cloud hybrid systems, *International Workshop on Cloud Management*, Osaka, Japan, 2010, pp. 343–349.
58. G. Xie, Y. Chen, R. Li and K. Li, Hardware cost design optimization for functional safety-critical parallel applications on heterogeneous distributed embedded systems, *IEEE Trans. Ind. Inform.* **14** (2018) 2418–2431.
59. G. Xie, J. Jiang, Y. Liu, R. Li and K. Li, Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems, *IEEE Trans. Ind. Inform.* **13** (2017) 1068–1078.
60. J. J. Durillo and R. Prodan, Multi-objective workflow scheduling in Amazon EC2, *Cluster Comput.* **17** (2014) 169–189.
61. J. J. Durillo, H. M. Fard and R. Prodan, MOHEFT: A multi-objective list-based method for workflow scheduling, *CloudCom* **61** (2012) 185–192.
62. J. J. Durillo, R. Prodan and J. G. Barbosa, Pareto tradeoff scheduling of workflows on federated commercial clouds, *Simul. Model. Pract. Theory* **58** (2015) 95–111.
63. N. D. Man and E. N. Huh, Cost and efficiency-based scheduling on a general framework combining between cloud computing and local thick clients, *Int. Conf. Computing*, 2013, pp. 258–263.
64. S. Su, J. Li, Q. Huang, X. Huang and K. Shuang, Cost-efficient task scheduling for executing large programs in the cloud, *Parallel Computing* **39** (2013) 177–188.
65. J. Li, S. Su, X. Cheng, Q. Huang and Z. Zhang, Cost-conscious scheduling for large graph processing in the cloud, *IEEE Int. Conf. High Performance Computing and Communications*, 2011, pp. 808–813.
66. A. Verma and S. Kaushal, A hybrid multi-objective particle swarm optimization for scientific workflow scheduling, *Parallel Computing* **62** (2017) 1–19.
67. O. Udomkasemsub, X. Li and T. Achalakul, A multiple-objective workflow scheduling framework for cloud data analytics, *Int. Joint Conf. Computer Science and Software Engineering*, 2012, pp. 391–398.
68. H. Chen, J. Zhu, Z. Zhang, M. Ma and X. Shen, Real-time workflows oriented online scheduling in uncertain cloud environment, *J. Supercomput.* **4** (2017) 1–17.
69. I. Casas, J. Taheri, R. Ranjan, L. Wang and A. Y. Zomaya, A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems, *Future Gener. Comput. Syst.* **74** (2017) 168–178.
70. G. Yao, Y. Ding, Y. Jin and K. Hao, Endocrine-based co-evolutionary multi-swarm for multi-objective workflow scheduling in a cloud system, *Soft Comput.* **21** (2017) 1–14.

71. C. Szabo and T. Kroeger, Evolving multi-objective strategies for task allocation of scientific workflows on public clouds, *Mon. Not. R. Astron. Soc.* **425** (2012) 2824–2839.
72. Z. Zhu, G. Zhang, M. Li and X. Liu, Evolutionary multi-objective workflow scheduling in cloud, *IEEE Trans. Parallel Distrib. Syst.* **27** (2016) 1344–1357.
73. Z. Wu, X. Liu, Z. Ni, D. Yuan and Y. Yang, A market-oriented hierarchical scheduling strategy in cloud workflow systems, *J. Supercomput.* **63** (2013) 256–293.
74. A. Khalili and S. M. Babamir, Optimal scheduling workflows in cloud computing environment using pareto-based grey wolf optimizer, *Concurrency Comput. Pract. Exp.* **29**(11) (2017).
75. W. A. Tan, Y. Sun, L. X. Li, G. Z. Lu and T. Wang, A trust service-oriented scheduling model for workflow applications in cloud computing, *IEEE Syst. J.* **8** (2014) 868–878.
76. X. Ye, S. Liu, Y. Yin and Y. Jin, User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm, *Knowl.-Based Syst.* **135** (2017) 113–124.
77. H. M. Fard, R. Prodan, J. J. D. Barrionuevo and T. Fahringer, A multi-objective approach for workflow scheduling in heterogeneous environments, *IEEE/ACM Int. Symp. Cluster, Cloud and Grid Computing*, 2012, pp. 300–309.
78. K. Bousselmi, Z. Brahmi and M. M. Gammoudi, DR-SWDF: A dynamixally reconfigurable framework for scientific workflows deployment in the cloud, *Scalable Comput.* **18** (2017) 177–193.
79. H. Ji, W. Bao and X. Zhu, Adaptive Workflow scheduling for diverse objectives in cloud environments, *Trans. Emerg. Telecommun. Technol.* **28**(2) (2017).
80. S. S. Rajan, Why Thick Clients Are Relevant in Cloud Computing, http://cloudcomputing.sys-con.com/node/1694221.
81. A. Verma and S. Kaushal, Cost-time efficient scheduling plan for executing workflows in the cloud, *J. Grid Compu.* **13** (2015) 1–12.
82. X. Zhang, Y. Tian and Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, *IEEE Trans. Evol. Comput.* **19** (2015) 761–776.
83. S. J. J. Chen, C. L. Hwang, M. J. Beckmann and W. Krelle, *Fuzzy Multiple Attribute Decision Making: Methods and Applications*, Vol. 375 (Springer-Verlag, 1994), pp. 302–310.
84. P. Liu and X. Zhang, Research on the supplier selection of a supply chain based on entropy weight and improved ELECTRE-III method, *Int. J. Prod. Res.* **49** (2011) 637–646.